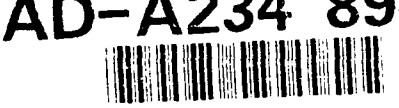


(2)

RADC-TR-90-404, Vol XV (of 18)
Final Technical Report
December 1990

STRATEGIES FOR COUPLING SYMBOLIC AND NUMERICAL COMPUTATION IN KNOWLEDGE BASE SYSTEMS

Northeast Artificial Intelligence Consortium (NAIC)

Minsoo Suk

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

This effort was funded partially by the Laboratory Director's fund.

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Services (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-90-404, Volume XV (of 18) has been reviewed and is approved for publication.

APPROVED: Sharon M. Walter

SHARON M. WALTER
Project Engineer

APPROVED: Raymond P. Urtz Jr.

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER: B G Oaks

BILLY G. OAKS
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COES) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1990	Final Sep 84 - Dec 89

4. TITLE AND SUBTITLE
STRATEGIES FOR COUPLING SYMBOLIC AND NUMERICAL COMPUTATION IN KNOWLEDGE BASE SYSTEMS

5. FUNDING NUMBERS
C - F30602-85-C-0008
PE - 62702F
PR - 5581
TA - 27
WU - 13
(See reverse)

6. AUTHOR(S)
Minsoo Suk

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Northeast Artificial Intelligence Consortium (NAIC)
Science & Technology Center, Rm 2-296
111 College Place, Syracuse University
Syracuse NY 13244-4100

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Air Development Center (COES)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RADC-TR-90-404, Vol XV (of 18)

11. SUPPLEMENTARY NOTES (See reverse)
RADC Project Engineer: Sharon M. Walter/COES/(315) 330-3577

This effort was funded partially by the Laboratory Director's fund.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Northeast Artificial Intelligence Consortium (NAIC) was created by the Air Force Systems Command, Rome Air Development Center, and the Office of Scientific Research. Its purpose was to conduct pertinent research in artificial intelligence and to perform activities ancillary to this research. This report describes progress during the existence of the NAIC on the technical research tasks undertaken at the member universities. The topics covered in general are: versatile expert system for equipment maintenance, distributed AI for communications system control, automatic photointerpretation, time-oriented problem solving, speech understanding systems, knowledge base maintenance, hardware architectures for very large systems, knowledge-based reasoning and planning, and a knowledge acquisition, assistance, and explanation system.

The specific topic for this volume is coupling symbolic and numerical computation as an effective means of solving problems in Knowledge Base Systems.

14. SUBJECT TERMS
Artificial Intelligence, Knowledge Bases, Symbolic Computation, Constraint Propagation

15. NUMBER OF PAGES
80

16. PRICE CODE

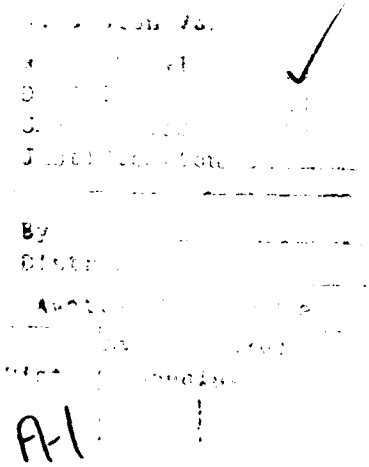
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Block 5 (Cont'd) Funding Numbers

PE - 62702F	PE - 61102F	PE - 61102F	PE - 33126F	PE - 61101F
PR - 5581	PR - 2304	PR - 2304	PR - 2155	PR - LDFP
TA - 27	TA - J5	TA - J5	TA - 02	TA - 27
WU - 23	WU - 01	WU - 15	WU - 10	WU - 01

Block 11 (Cont'd)

This effort was performed by the Syracuse University, Office of Sponsored Programs.



CONTENTS

15.1. INTRODUCTION TO COUPLED SYSTEMS

It is widely recognized that coupling symbolic and numerical methods is an effective means of solving many problems in science, engineering and business. In order to solve various problems, both insight and precision are very frequently needed. Coupled systems guarantee the integration of the precision of traditional numerical computing with the problem solving and result explaining capabilities of symbolic processing. A second major reason for coupling is the need for more powerful and useful tools capable of overcoming the limitations of traditional tools, which are not coupled systems. Integrating formal numerical methods with methods based on symbolic knowledge is believed to be the key to the development of computing tools that can solve some of the problems currently known to be difficult to solve. Coupled systems link symbolic and numerical computing in a manner not found in conventional expert or knowledge-based systems. For coupled systems, effective problem solving must have some knowledge of the numerical processes embedded within them and must be able to reason about the application or results of those numerical processes. In general, the symbolic process of a coupled system is the top-level process controlling the numerical processes. However, the possibility of a numerical process controlling symbolic processes cannot be neglected, although, numerical algorithms are only abie to procedurally invoke symbolic processes.

15.1.1. Phases of Coupling in a Typical Problem-Solving Mechanism

15.1.1.1. Layers of Coupling

A typical knowledge-based system can be described in terms of four major component modules as shown in Figure 1. These modules are the following:

1) Domain independent or weak heuristics.

2) Model-directed heuristics guided by a qualitative model.

3) Quantitative model of the problem domain.

4) Planner which controls the problem-solving process.

In this section we describe the nature of coupling between various component modules. The interaction between modules occurs during specific phases of the problem-solving process. By decomposing the coupling process into its component layers we effectively study the nature of coupling during each problem-solving phase.

15.1.1.2. Coupling of Weak Heuristics with a Qualitative Model

In most knowledge based systems the initial phase of problem solving is largely data-directed. The system does not know which model or set of models to invoke a priori. In such a situation the system resorts to weak heuristics, i.e. heuristics which do not make strong assumptions about the problem domain. During the course of problem solving the system notices certain "interesting patterns" in the data, giving the system clues as to which model or set of models to invoke in order to speed up the problem-solving process.

In a typical searching problem the system could, in the initial phase, use weak techniques such as A* or means-ends analysis and during the course of problem solving switch to more powerful domain-specific techniques for guiding the search process. In such

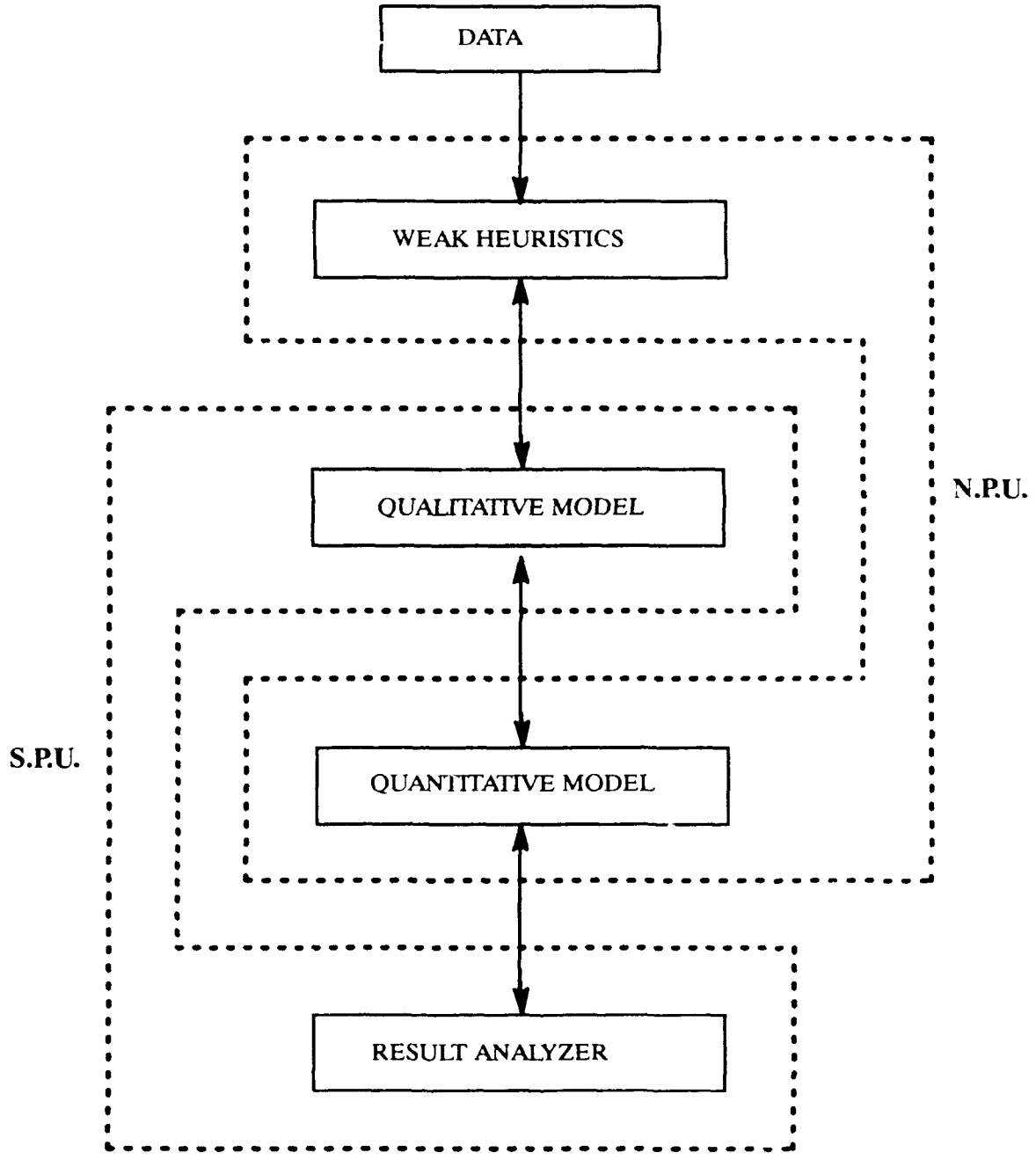


FIGURE 1. Block Diagram of Typical Coupled System

systems this phase of coupling could be described as the coupling of weak heuristics with a qualitative model of the problem domain. In such systems it is not feasible to couple the weak heuristics directly with a detailed quantitative model. Some qualitative understanding of the model is necessary. This qualitative understanding of the quantitative model is represented in the form of a qualitative model.

15.1.1.3. Coupling of a Qualitative Model with a Quantitative Model

In many scientific/engineering applications a mere qualitative understanding of the problem is not enough. A detailed quantitative model has to be invoked for the precise computation of certain parameters associated with the problem under consideration. The invocation of the model has to be done based on some evidence gathered by the qualitative model. In most applications the most intensive phase of the problem-solving process, in terms of computational resources, follows the invocation of a quantitative model. It is important, therefore, that the appropriate quantitative model be invoked. The efficacy of coupling in this phase is largely governed by how closely the qualitative model reflects the underlying quantitative model. The qualitative model should thus represent just enough knowledge about the quantitative model so as to ensure that the quantitative model is appropriately and unambiguously invoked. The coupling of qualitative model with a detailed quantitative model thus represents that phase of problem solving which involves precise computation of numerical parameters.

15.1.1.4. Coupling of the Quantitative Model with the Result Analyzer

In most engineering and scientific applications it is necessary to not only obtain precise results but also interpret them intelligently and offer qualitative explanations. Further,

it is necessary to detect cases where the quantitative model has failed to give a satisfactory solution and suggest alternatives. Thus, the function of the result analyzer is to interpret the results of the numerical procedures, offer qualitative explanations, and guide the problem-solving process through possible alternatives in the event of failure. Control of the problem-solving process is thus highly dependent on the result analyzer.

In this section we have considered the various layers of coupling. Each layer of coupling represents a definite phase in the problem-solving process and greatly affects the overall system performance.

15.1.2. Coupling Strategies

The control of the problem-solving process is an essential feature of every knowledge based system. The coupling strategy employed directly affects the flow of control in the various phases of the problem-solving process. Opportunistic coupling and model-driven strategy are two major coupling strategies found in most knowledge-based systems.

15.1.2.1. Opportunistic Coupling (Data-directed or Bottom-up)

In a purely data-directed mode of coupling, the flow of control is strictly bottom-up through the various phases of problem solving. Each phase of problem solving is invoked in sequence until a solution has been reached. A typical data-directed coupling strategy could be described in the following manner:

1) Hypotheses are formed based on initial evidence present in the data.

2) Hypotheses aggregation is done based on certain rules of inference.

3) Appropriate qualitative model is invoked when there is enough evidence.

4) Based on the qualitative explanation offered by the qualitative model, appropriate mathematical model is invoked.

5) The results are verified. If successful, stop. Else go to (2) and use an alternative set of rules or go to (1) to consider alternative hypotheses.

15.1.2.2. Model-driven Strategy (Top-down)

The model-driven strategy is essentially a hypothesize-and-test strategy. Blind hypothesize-and-test strategies do not use the data intelligently. When the number of possible models is very large, this method tends to get exhaustive. Model-driven strategies could be refined to a certain extent by

1) Employing a hierarchical matching procedure in which the models are hierarchically configured from coarse to finer levels of detail.
2) Models could be invoked in an orderly fashion depending on cost, likelihood and importance.
3) Hypothesize-and-test strategies could be carried out in parallel as recommended by Fishler [1].

However, this refinement does not overcome the basic shortcoming of a purely model-driven strategy completely, i.e the need to make intelligent use of the available data.

Both model-driven and data-directed strategies in their simplistic forms do not pose challenging issues from the point of view of a formal study of coupling. Apart from being practically unfeasible for many problems, the nature of interaction between the various phases of problem-solving is predetermined. Thus, in order to achieve more meaningful control over the problem-solving process, hybrid strategies for coupling are needed. A hy-

brid strategy that combines both data-directed and model-driven strategies would have to intelligently use both sources of knowledge: knowledge present in the raw data and knowledge present in the stored models (or rules). The coupled systems WX1[2], STAR-PLAN[3], PMS-1[4] are good examples of hybrid systems which can be classified broadly as deep-coupled systems.

15.2. CATEGORIZATION OF COUPLED SYSTEMS

15.2.1. Categorization based on Shallow vs. Deep Coupling

Coupled systems are generally categorized into shallow and deep coupled systems according to the amount of their knowledge about the involved numerical processes. Shallow-coupled systems have limited knowledge of the involved numerical processes and treat numerical routines as black boxes. Deep-coupled systems utilize extensive knowledge of each numerical process and provide an intelligent interface to numerical routines. The knowledge of each process is integrated with other information and used directly by the knowledge-based system, i.e. the symbolic processes, during problem solving.

In distributed problem solving, the strategies for coupling could be cooperative or competitive, i.e. the numerical processes could exhibit cooperative or competitive behavior. In a competitive mode, several numerical processes solve the same problem independently of each other. In a cooperative environment they are coupled in a manner such that they solve the same problem more efficiently as compared to a single numerical process. Cooperative coupling in a distributed environment enhances the overall performance, especially when complex problems are being solved. In order to achieve deep coupling, the involved processes must be coupled cooperatively, i.e. they should be capable of solving a subproblem and communicating with each other to share raw or processed data. The coupled system thus needs to incorporate a deep understanding of the interacting processes in order to achieve deep coupling which in turn provides robustness and good performance.

Deep-coupled systems however, entail a substantial overhead in terms of design since the knowledge about the interacting processes has to be carefully engineered. In the absence of this knowledge one has to resort to a competitive mode of coupling in which each process solves the problem independently hoping that one of the processes will come up with

a solution in a reasonable length of time. Competitive modes of coupling tend to be wasteful in terms of resource consumption.

15.2.2. Categorization based on Architectures for Coupling

Architectures for coupling could be broadly classified into two categories, centralized and distributed architectures.

15.2.2.1. Centralized Architecture

In a centralized architecture, the control of problem solving is done exclusively by a centralized planner. The planner decides on the mode of coupling between the problem-solving modules during various phases of the problem-solving process. The centralized architecture is easy to design and analyze since all the control information is located in a single module. However, since each problem-solving module needs to interact with the planner, communication with the planner proves to be a major bottleneck. The failure of the planner would thus be catastrophic, making the system less robust.

It is also possible for the centralized architecture to be hierarchically configured. This means :

1) The planner is a hierarchical planner.
2) The numerical processes could be structured in a hierarchical manner in an increasing magnitude of precision or sophistication.
3) The symbolic processes are hierarchically structured in an increasing magnitude of detail.

Typically, a hierarchical architecture would give the planner a better control of the problem-solving process since problem solving can then proceed in stages of coarse to fine

degrees of refinement.

15.2.2.2. Distributed Architecture

As opposed to the centralized approach, the problem-solving system could also be configured as a society of distributed independent agents, each agent with an independent planner. The qualitative models and also the quantitative models could be treated as independent agents with local planning capabilities. Such a distribution would be an object-centered distribution.

The choice of architecture is ultimately governed by the overall goals of the system designer. Centralized architectures are easier to design and analyze. However since the control information is localized within a single planner module, the other problem-solving modules need to interact with the planner very heavily. This could cause a communication bottleneck. In a distributed architecture, the design issues such as

1) nature of interactions between agents
2) the means of communication between agents and
3) stability of the system and guaranteed convergence to a solution

have to be dealt with. The advantages of a distributed architecture are greater fault tolerance to failure and exploitation of parallelism in the problem. In some problems, however, the nature of distribution is decided by the very definition of the problem.

15.3. MODELING COUPLED SYSTEMS

In this section a general model of coupled systems is described. This model would help to analyze and understand the structure of coupled systems. This model is fairly general and can be used in building new systems. It also allows incremental development since the structure of a coupled system is organized into three distinct units having their own intrinsic characteristics. We will focus on the modeling of deep-coupled systems, since the coupling mechanism in shallow-coupled systems is fairly straightforward.

15.3.1. A General Model of Coupled Systems

The proposed model consists of three parts, a Numerical Processing Unit (NPU), a Symbolic Processing Unit (SPU) as shown in Figure 1 and a Coupling Unit (CU), which is not shown explicitly in Figure 1.

The Numerical Processing Unit consists of a set of numerical processes which accomplish specific tasks via numerical means. The Symbolic Processing Unit consists of a set of symbolic processes needed for problem solving as well as to reason about the application of results of numerical processes. Four major component modules of a coupled system are mentioned in the previous sections. Two of those components, the domain independent heuristics and the qualitative model, are parts of the SPU. The quantitative model of the problem domain is in the NPU. The inclusion of the planner in a specific unit is case dependent. In a distributed architecture, the planner is a part of the SPU and each agent (which in most cases, is an expert system) has its own planner. In most cases of centralized architectures the planner is also included in the SPU, irrespective of whether the problem-solving strategy is top-down or bottom-up. When the SPU consists of an expert system, the planner

which controls the problem-solving process, is in the knowledge base of the expert system (in most cases a rule-based expert system). The example that the planner belongs to CU can be seen in "Incremental Construction of Coupled Systems" in section 15.4.1.3.

15.3.2. The Basic Functions of the Coupling Unit

The coupling or interaction between the NPU and the SPU is controlled by the Coupling Unit (CU). Four basic functions of the CU are described below.

(1) Management of Communication

The communication between processes needs to be fully asynchronous since numerical and symbolic processes need to exchange information with each other and to invoke each other during the problem-solving process. For example, symbolic processes need to transfer the information about the selected numerical process, transfer problem specification, and send requests for invocation with the requisite parameters to the numerical processes. Similarly, numerical processes need to transfer the result of computation to symbolic processes. In the case of distributed problem solving, a big complex problem would be divided into several subtasks. The processes, especially in the SPU, need to exchange information to solve their own subtasks. Therefore, it is important for the CU to fully control the asynchronous communication between SPU and the NPU, and between communication agents in the SPU or the NPU.

J.S. Kowalik [38] mentions the *blackboard architecture* as being a very popular choice for the overall system architecture for coupled systems [3-6]. Among the advantages of this approach is the ability to stratify the problem solving and process knowledge. This allows all problem solving and other meta-level information to be uniformly represented in the black-

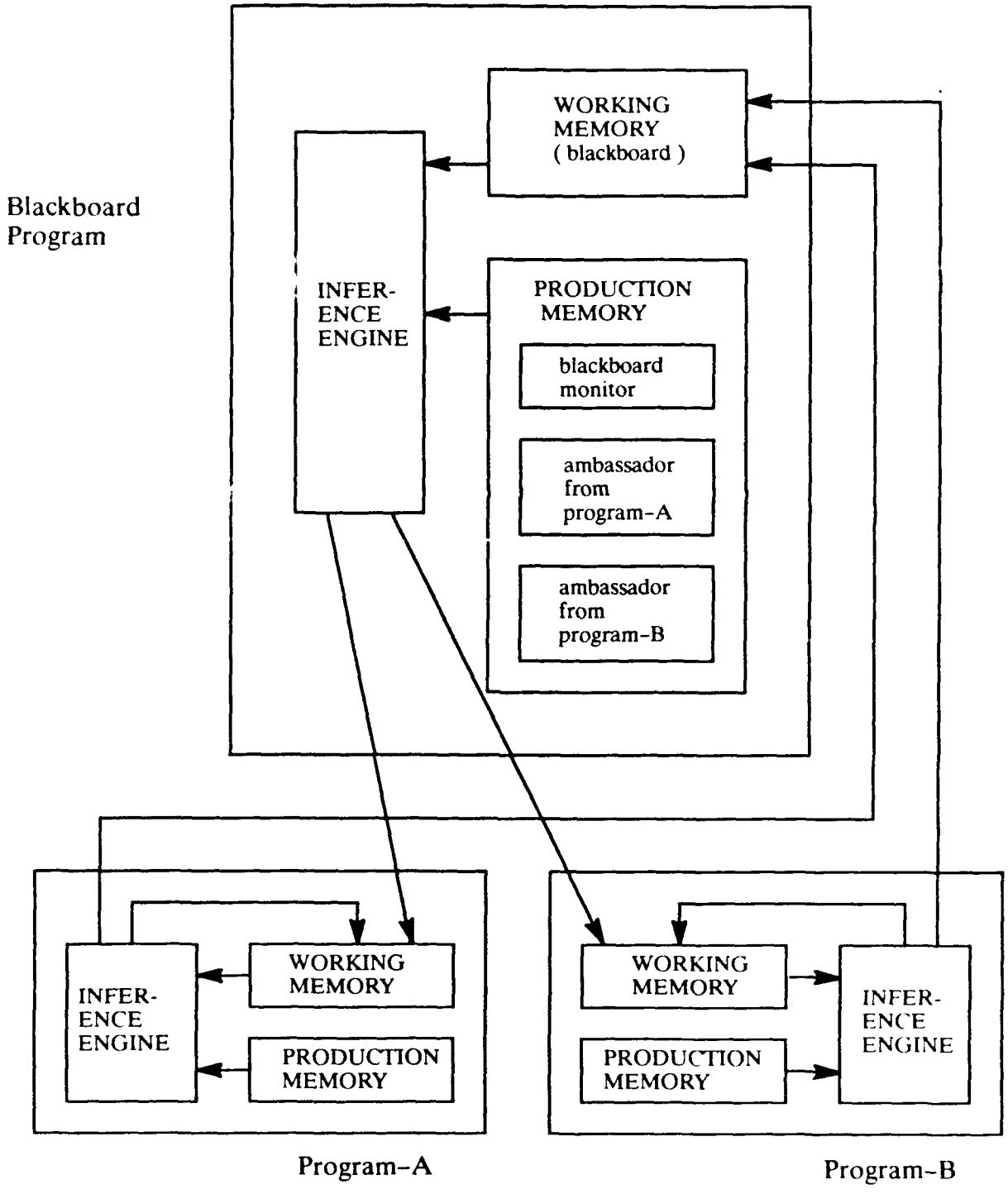


FIGURE 2. Example of the Blackboard in COPs

board monitor. Individual symbolic and numerical processes can be incorporated as separate multi-level knowledge sources. All information specific to an individual process is represented within the corresponding knowledge source. The blackboard monitor manages communication and is a part of the Coupling Unit. Figure 2 shows a good example of the *blackboard architecture* in COPS [5].

(2) Scheduling

The scheduling task includes allocation/deallocation of the subtasks to the numerical and symbolic processes, activation of the processes selected by the SPU. collection of the results from the processes and initiation of user interfacing or external system interfacing.

(3) Translating

The CU translates messages from different processes if necessary and converts high-level specification to low-level specifications and vice versa. When the coupling strategy is model-driven, a high-level specification means a symbolic representation of information whereas a low-level specification means numerical representation of parameters extracted from the high-level specification necessary for numerical analysis. In coupled systems adopting data-directed strategies, the high-level specification is the description of a higher level object, which is formed by combining lower-level objects in a bottom-up approach. The lower-level objects can be characterized by low-level specification from numerical processing of raw data.

(4) User Interface

The CU provides the user interface by receiving the problem specification as its input. It also provides means for explaining the results or reporting current processing status to the user with linguistic/graphic descriptors. It helps to make the system transparent to the users.

15.3.3. Communication between SPU and NPU

Communication between processes in a coupled system is considered an important problem because problem solving proceeds by exchanging messages between symbolic processes and numerical processes. As it has been shown before, the symbolic processing unit and the numerical processing unit communicate with each other generally by fully asynchronous message passing. These messages can be requests for some processes to initiate the processing of the data, parameters necessary for the processing of data and results returned after the processing. In most cases symbolic processes are more active and mainly ask for number crunching jobs or for some information which they need during the problem-solving process from the numerical processes. It is the symbolic processes which usually control the communication in a coupled system.

A coupled system is generally implemented on a heterogeneous environment. For example, the coupled system can be composed of two different environments – a LISP machine and a general purpose computer. This causes somewhat serious difficulties in communication between processes implemented in a heterogeneous environment.

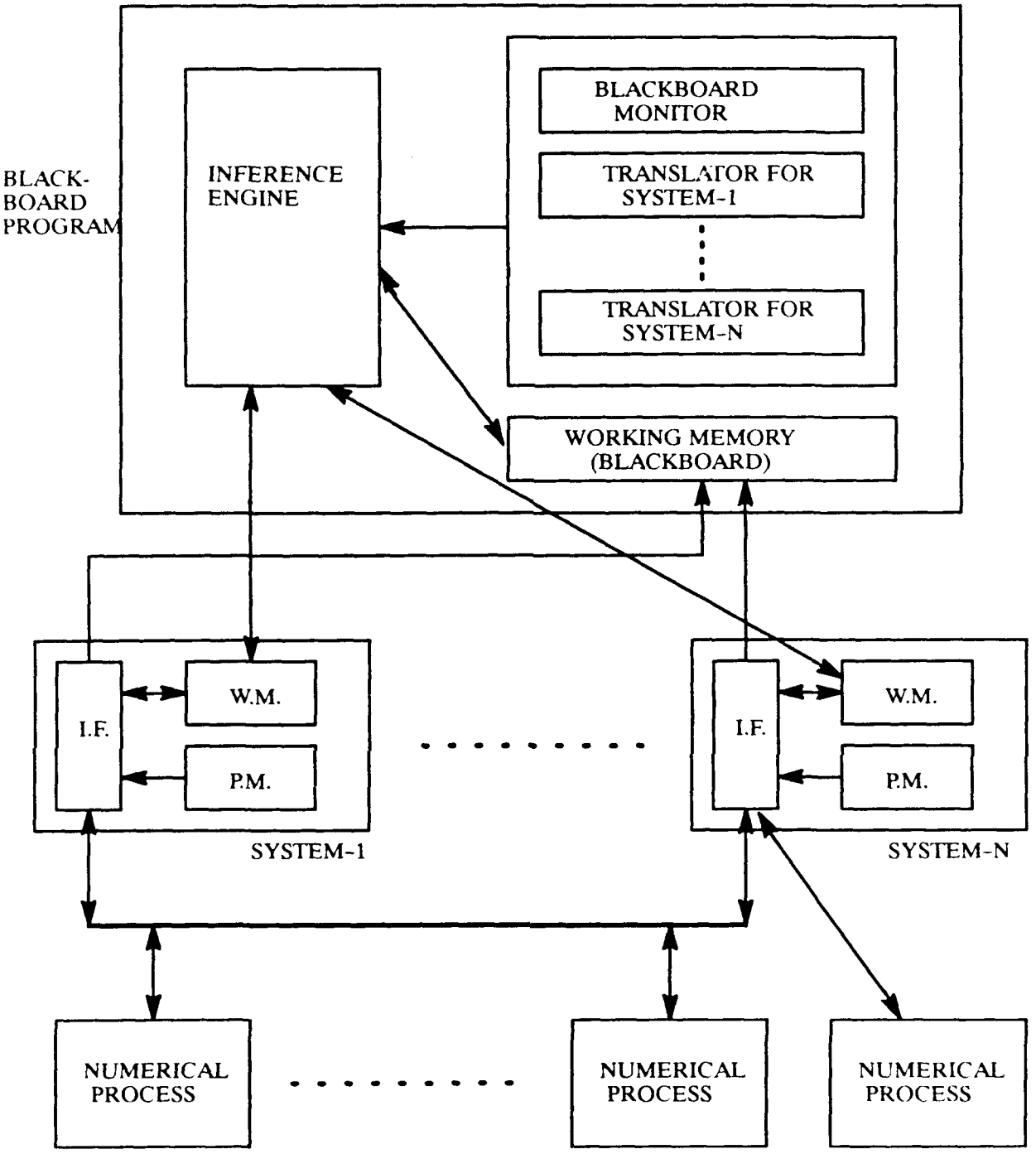
15.3.3.1. Simple Coupling

Most simple coupled systems are composed of one symbolic process and several numerical processes. In simple coupling, the message passing scheme is a kind of remote procedure call. The content of a message is a set of parameters. The symbolic process which issues a message becomes a client and waits for a reply from the numerical processes. The numerical processes act like servers.

Simple coupling takes place when the symbolic process calls the numerical processes and receives replies during the course of either bottom-up or top-down problem solving. Even if the simple system is implemented in an heterogeneous environment, for example an expert system and a general purpose computer, these remote procedure calls could still be simple because the content of the message is limited to parameter passing between the symbolic process as a client and the numerical processes as servers.

15.3.3.2. Coupling in the Distributed System

As the size of the coupled system increases, the number of numerical processes and symbolic processes increases. Some numerical processes serve a typical symbolic process or all numerical processes can be invoked by any symbolic process. This is not a major problem since simple messages have to be communicated. However communication can also take place between the symbolic processes. If a coupled system is implemented as a heterogeneous distributed system, then a link has to be provided between every pair of systems within the distributed system. This link should translate the data structures of the sending process to that of the receiving process if the two processes are incompatible. The high cost of links makes this approach expensive to implement.



I.F.: INFERENCE ENGINE
W.M.: WORKING MEMORY
P.M.: PRODUCTION MEMORY

FIGURE 3. Blackboard Architecture

An inexpensive alternative is to build a central facility for collecting and distributing messages. The central facility can be a blackboard. The sending process writes messages on the blackboard and the receiving process reads message from it. Thus only a link between each system and the blackboard is necessary. Each link has to provide just the means to translate the data structure of each system to that of the blackboard data base. This *blackboard architecture* is shown in Figure 3.

15.3.4. Desirable Environment for Implementation of Coupled Systems

The present coupled systems implemented in heterogeneous environments work slowly because messages have to be translated into the common data structure. For example, in the SIMPOP coupled system [7], the NPU is implemented in Fortran-77 while the SPU is implemented in another environment using Prism. Prism is compatible with Franz LISP but incompatible with Fortran. As a result message passing is routed via a user interface. The implementation uses disk files which provide the common data structure. This process of translating the data structure via diskfiles needs frequent intervention of the operating system thereby slowing the system considerably.

It is considered a good strategy to use conventional expert systems to implement coupled systems rather than to build new symbolic processing systems. The systems implemented in either a LISP or a Prolog environment can be used for symbolic processing. Languages like Fortran or C, which have functional properties, could be used for numerical processing. Interfacing dissimilar languages in a coupled system composed of different languages, however, is a very complex problem from an implementation point of view.

15.3.4.1. Abstract Data Types

If the system requires that the data be shared by several sub-systems implemented in different languages, *abstract data types* are needed because they provide data hiding and relieve the user from implementation details. *Object-Oriented Programming* (O.O.P.) languages provide the *abstract data types* that can be shared by sub-systems implemented in different languages.

There are two ways by which coupled systems can be implemented more easily with O.O.P. languages than with conventional procedural languages and functional languages. One approach is to implement the entire coupled system in one O.O.P. language that supports *abstract data types* in order to avoid frequent intervention of the operating system. The other way is to have as many identical *abstract data types* as the languages used. In the case *abstract data types* talk to each other during communication between different processes implemented with different languages. It is important to maintain coherence among the *abstract data types*.

15.3.4.2. Concurrence

If the coupled system was implemented in one O.O.P. environment, then concurrence in the environment would be desirable. If there are many distributed processes, some processes can run concurrently in order to reduce running time or others should run concurrently because of their interrelationships. Ideally, an O.O.P. language with the properties of a *concurrent programming* language is desirable in this case.

Message passing is an important aspect of *concurrent programming*. There are two kinds of message passing schemes --- asynchronous communication and synchronous communication. When using asynchronous communication, a process can send a message

and continue. This message may be picked up by another process at a later time. In contrast, when using synchronous communication the two processes must send and receive the messages simultaneously. If the sender waits only until the message is received before resuming, this is referred to as a synchronous send. While a synchronous send greatly simplifies the requirements for message buffering, it is difficult to be justified from a practical point of view. Although, on resumption, the sender can assume the message has been received, this does not guarantee that the transmitted information will be correctly processed, i.e. a process does not know whether a message has arrived at its destination after the send operation is completed. If a guarantee of reception is required, the sender must continue to wait for a separate acknowledgment explicitly sent by the receiver. If the guarantee is not required, the sender need not wait for the message to be received. This leads to the other form of communication, the asynchronous send, in which the sender resumes immediately after sending the message. Communication based on an asynchronous send is referred to as asynchronous message passing.

Another important aspect of the communication structure of a programming language for distributed systems is the technique used for identifying the sender/receiver of a message. This can be done explicitly by both processes (symmetric naming). More commonly, the sender names the receiver while the receiver is willing to communicate with any sender (asynchronous naming). This anonymous sender approach is suitable for a client/server relationship, particularly in the context of a remote procedure call where the identity of the caller is implicitly retained for the return. The server simply performs a function: it does not have to know who the client is.

15.3.4.3. System Integration

Implementing deep-coupled systems needs integration of processes which have different characteristics even in the same software system. As the system is expanded integration of several different runtime environments arising due to different programming languages has to be considered. One has also to handle the diversity problems when implementing coupling systems by integrating several different environments.

15.3.4.4. Incremental Development

Coupled systems need to keep up with the state of the art, which means they need to be incremented. For example, sometimes one may need to expand the problem domain, come up with new numerical algorithms and new A.I. techniques, and increment the knowledge-base in the symbolic processing unit. Thus the implementation of the coupled system has to guarantee incremental development so that the system can be easily expanded.

15.3.4.5. Overview of Object-Oriented Programming

Most programming languages support the "*data-procedure*" paradigm. Active procedures act on the passive data that is passed on to them. Object-oriented languages employ a data or *object-centered approach* to programming. Instead of passing data to procedures, one asks objects (data) to perform operations on themselves.

Object-oriented programming exhibits four characteristics [8].

(1) *Information Hiding*: The state of a software module is contained in private variables, visible only from within the scope of the module.

(2) *Data Abstraction*: A programmer defines an abstract data type consisting of an internal representation plus a set of procedures used to access and manipulate the data.

(3) *Dynamic Binding*: The object-oriented approach pushes the responsibility for some operation onto the objects themselves. This is known as "*Polymorphism*" since the same message can elicit a different response depending on the receiver (object) [9].

(4) *Inheritance*: It enables the programmer to create new classes of objects by specifying the differences between an existing class and a new class instead of starting from scratch each time. It thus reuses the existing code.

15.3.4.6. Advantages of O.O.P. for Implementing Coupling Systems

(1) Integration on Single Software System

In the *object-oriented programming* paradigm, we can map both symbolic and numerical programs to classes of objects. These classes can communicate via message passing. Therefore using *object-oriented programming* paradigm is very helpful for implementing coupled systems.

(2) Dealing with Diversity [10]

In *object-oriented programming* one can perform operations by passing messages to the object. Thus there exists a great advantage for coupling several different types of environments implemented in different languages. This is so because one can treat them as separate objects, essentially at least one object per programming language, one needs to consider just a message-passing protocol between them. This is a very important problem. Powell and Cole [11] alleviate this problem by using LISP for numerical processing unit and MRS [12] for symbolic representation because the underlying data representation scheme for both LISP and MRS are the same.

(3) Reliability and Modularity

Information hiding ensures reliability and modularity of software systems by reducing interdependencies among software components. Data abstraction, a way of *information hiding*, is a frame-like structure which is advantageous for implementing the control schemes for particular coupled systems which use frames for knowledge representation.

(4) Incremental Development

All four characteristics of *object-oriented programming* give modularity to the software system, which is a necessary condition for incrementing the system. Dynamic binding enables one to develop an incremental software system. One can increment the system just by adding a new type of object which requires writing new procedures while not modifying the existing software. During that process, *inheritance* makes it possible to reuse the existing code and reduce the redundancy.

15.4. CASE STUDIES & IMPLEMENTATION EXAMPLES

15.4.1. The Case Studies of Coupled Systems

To justify the general model of coupling developed in section 15.3.1., we have examined several deep coupling systems and come up with four kinds of generic structures which can represent almost all coupling systems. The structures will be called prototype classes in the remainder of this report. The coupled systems can be classified into one of these four prototypes, based on the role and the internal structure of the SPU. Although the first two prototypes can be merged into one based on the system-level decision, they are however distinct since coupling strategies employed are different, i.e the roles and the internal structures of the SPUs are also different. The first one mainly employs a top-down (model-driven) approach whereas the second mainly employs a bottom-up (data-directed) approach. These prototype classes are explained in sections 15.4.1.1. through 15.4.1.4.

15.4.1.1. Prototype Class based on the Hypothesize-and-Test Paradigm (Top-down)

A typical example of this prototype class is a system composed of a rule-based expert system and a numerical analysis unit for test simulation.

The rule-based expert system receives the problem specification from the user through the coupler and it hypothesizes the cause of the problem. The coupler translates high-level description of the hypothesis to lower-level specification and transfers it to the numerical analysis unit. This activates the selected numerical process. It also extracts the necessary parameters for the expert system from the results of numerical analysis. The expert system makes another hypothesis according to the results from the numerical proc-

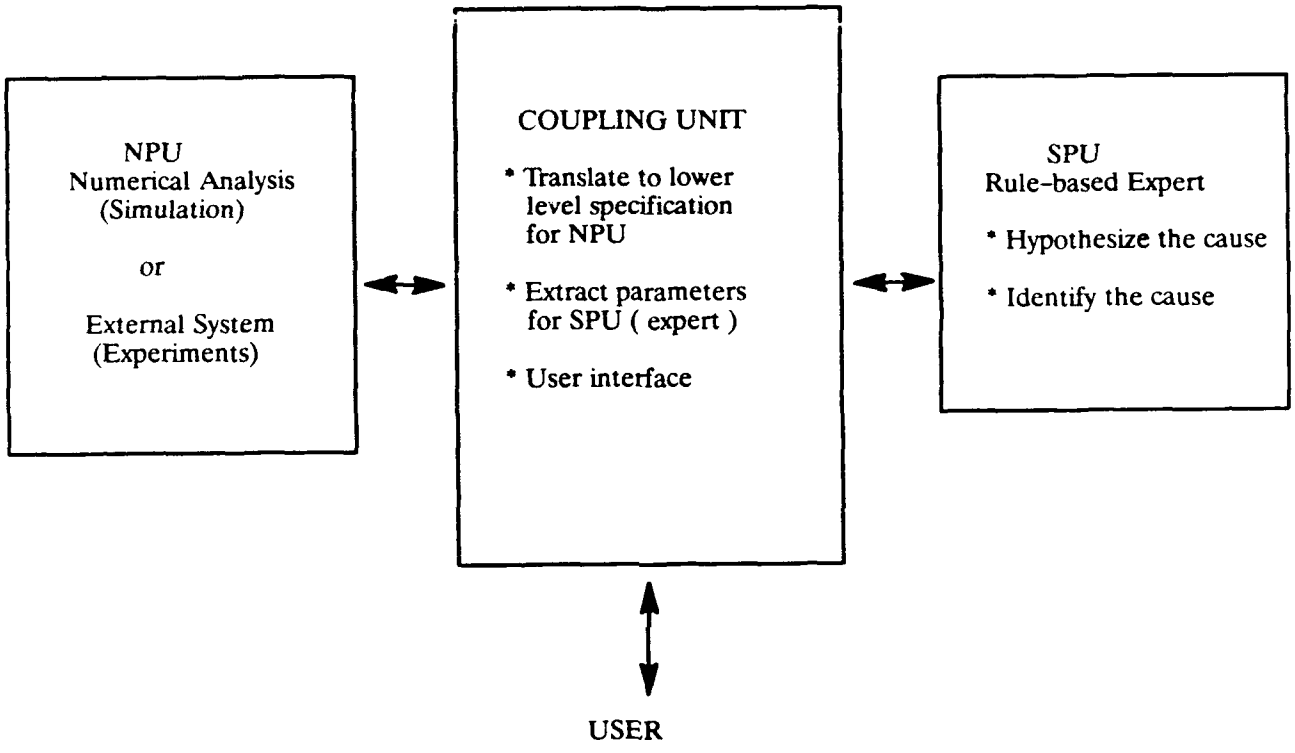


FIGURE 4. Model of Coupled System in Hypothesize-and-Test Paradigm

esses. Through this simple feed-back process, the expert system can identify the cause of the problem.

In this system, the rule-based expert system is the symbolic processing unit and the numerical analysis part is a numerical processing unit. We examined the systems mentioned in [13] and [14] and categorized them as belonging to this prototype as shown in the block diagram of Figure 4.

15.4.1.2. Coupling Expert Systems and Numerical Processing Units (Bottom-up)

As an example of this prototype class, a system is composed of a numerical processing unit (especially an image processing unit) and an expert system or distributed expert system as a symbolic processing unit.

In WX1 [2], the image processing unit gets low-level information from an external *doppler radar*. The image processing unit segments the image and creates an image database. This is essentially a bottom-up process. The expert system has a symbolic representation of the image and its features in its working memory, and production rules necessary for identifying the goal-object using a bottom-up approach in its production memory. The expert system requests the image processing unit for detailed information, i.e. properties of features and relationships between features. It then responds to the requests and creates a new higher-level feature by combining two or several features. This procedure creates a feature database both in the image processing unit and in a symbolic representation database in the working memory.

The coupler's role in this system is simply that of a user interface which shows the user its goal object identified through the procedure of fully asynchronous message passing between the image processing unit, an NPU, and the production-rule expert system, a SPU.

In the case of coupling distributed expert systems with an image processing unit [15], the symbolic processing unit is composed of vertically (different levels) and horizontally (at the same level) distributed expert systems. These expert systems communicate with each other through a blackboard. The image processing unit provides segmented image to the expert systems which can identify an object at several levels such as the class level and the object level. The reasoning level receives the results from these levels and reasons about the object. If the reasoning level can explain the result without contradiction then the coupler describes it to the user by a linguistic descriptor, otherwise it feeds back the cause of the contradiction to the image processing unit. In this feed-back process, the coupler also translates the information to a low-level specification and activates the selected numerical process (segmentation scheme) in order for the image processing unit to resegment the image.

The coupler, therefore, has the characteristics of a communication manager between the distributed expert systems, a scheduler which activates the selected segmentation modules (namely, knowledge-driven segmentation), a translator between reasoning level and the image processing unit, and a user interface which provides linguistic descriptions.

We also examined the SIMPOP system [7] that uses a simulation model with machine intelligence. It incorporates problem solving through heuristics and employs both qualitative and quantitative modeling. The system integrates mathematical simulation that imitates demo-graphic mechanics of biological populations, with rule-based reasoning that contains procedural knowledge and induces the simulations for examples and analysis. Either of them can access a statistical database. Those portions of the expert systems which deal with qualitative representation contain information of how to link to the existing numerical models as well as how to choose parameters for the particular module. Results of simulations are presented graphically, and the system requires operator intervention for guidance and con-

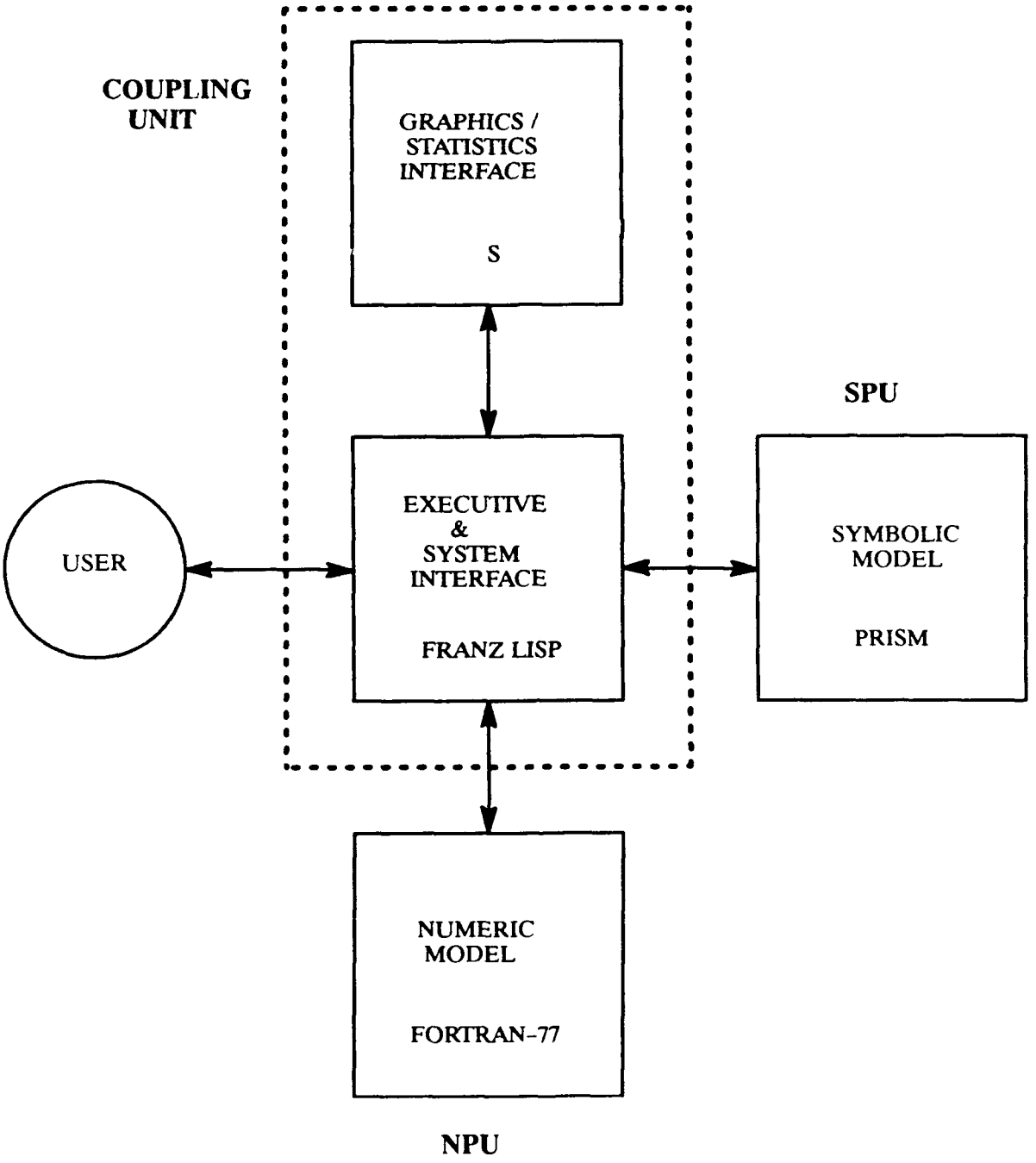


FIGURE 5. Block Diagram of SIMPOP

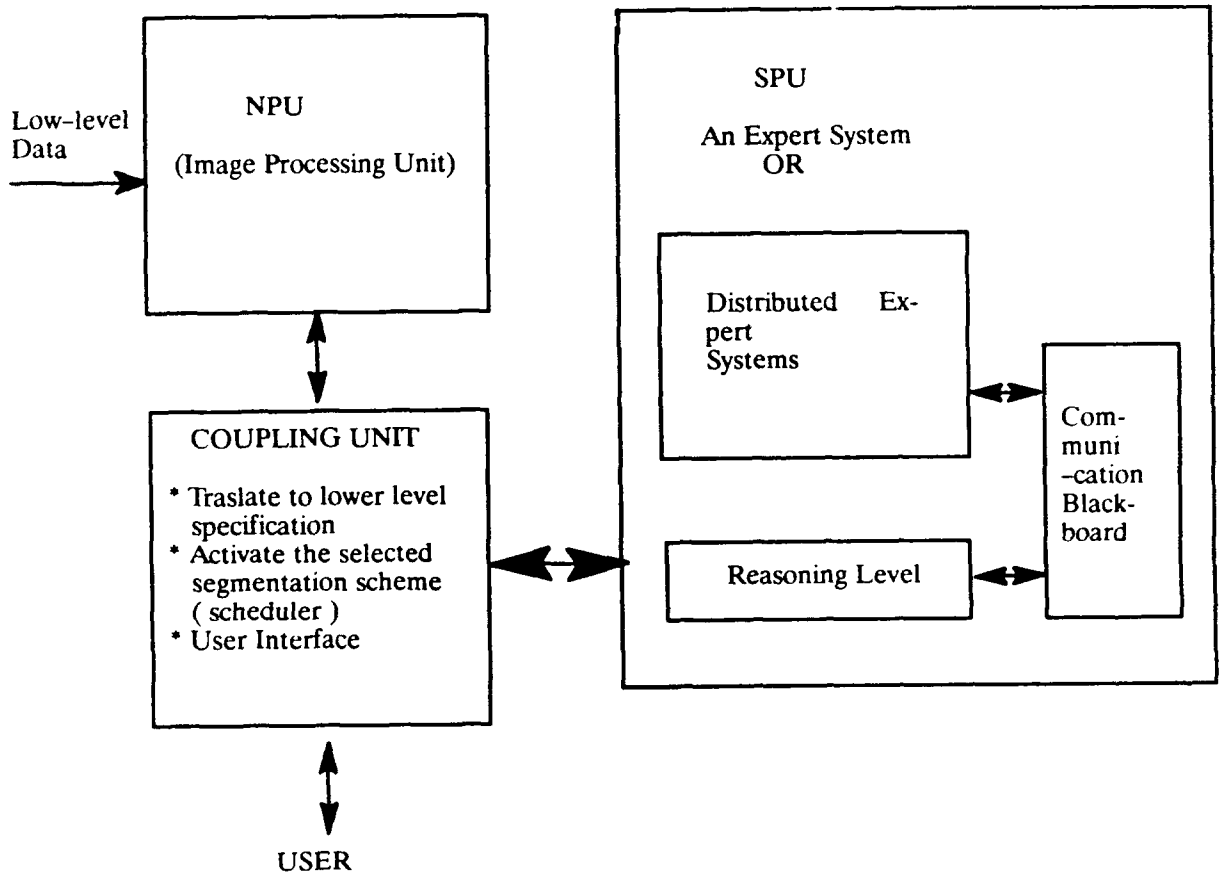


FIGURE 6. Model of Coupling Expert Systems and NPU (Bottom-up Approach)

trol. The global architecture is composed of an *object-oriented message passing architecture* and a *blackboard architecture*. The structure of SIMPOP is shown in Figure 5. The Floor Plan Generator [11] can also be categorized as belonging to this prototype class which is shown in Figure 6.

15.4.1.3. Incremental Construction of Coupled systems [16]

This prototype class represents the smooth upgrade of a shallow coupled, rapid prototyped system to a more intelligent, and more widely applicable, deep coupled system, without being subjected to frequent modification.

KIM (Knowledge-based Integration Manager) couples shallow coupled systems. KIM has frames about the type of modules - each module represents a shallow coupled system - and information about the translation between various classes of modules. It also has a blackboard for communication between modules and between modules and the scheduler, and a scheduler for monitoring the blackboard and scheduling modules. An explanatory component included in KIM provides system transparency to the user through a scheduler.

As shown in Figure 7, the scheduler and the explanatory component of KIM are the coupler which manages communication, user interface, and activates the selected modules. The remaining part of KIM, that is, the frame work of deep information about the shallow coupled systems, and modules which can activate them, is the symbolic processing unit. It also includes the symbolic processing parts of shallow coupled systems. The numerical processing unit is the global set of numerical processes which are coupled in the shallow coupled systems.

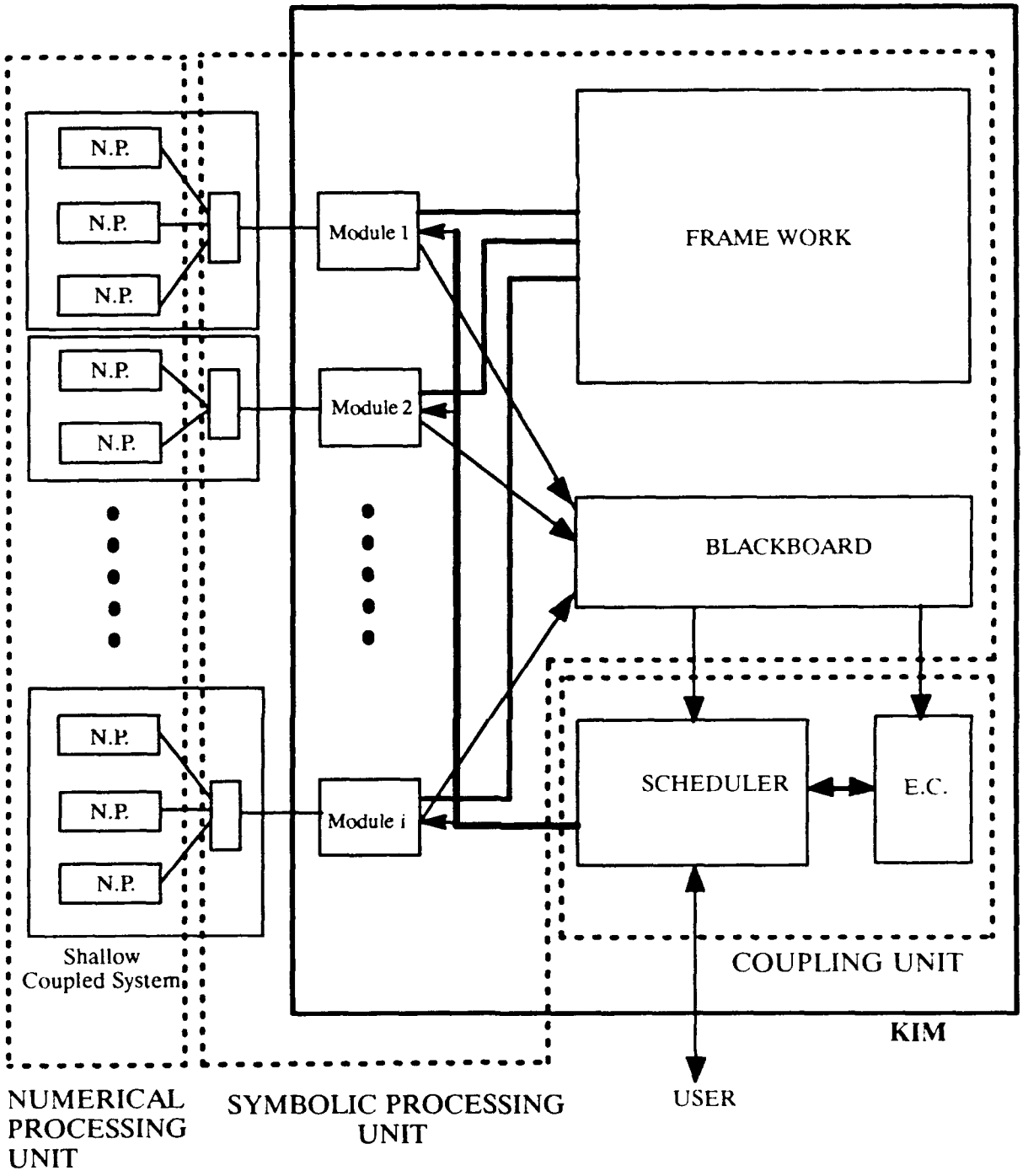


FIGURE 7. Model of Incremental Construction of Coupled Systems

E.C.:Explanatory Component
N.P.:Numeric Process

15.4.1.4. Coupling with Autonomous Communication Objects [17]

An example of this prototype class was implemented by Janos Sztidanovits and Byron Purves[17] at the Vanderbilt University. They define an Autonomous Communication Object (ACO) to be an extension of the *"object"* concept of object-oriented languages. ACOs are fully autonomous systems that can run, virtually or physically, in parallel. They can be dynamically allocated and may compete for the same resources. They communicate with each other by means of a fully asynchronous message passing protocol. They have also classified ACOs into two classes, Rule Network Object (RNO) and Procedural Network Object (PNO).

The RNOs can perform rule-based reasoning and can be allocated to the tasks which need symbolic processing. The PNO can be looked upon as a shallow-coupled system since it has selection rules, numerical processing units (especially for signal processing) and interface to the external systems. It can also perform numerical procedures. The message transfer and allocation/deallocation of ACOs to tasks are controlled by a communication manager.

As can be seen from Figure 8, this structure can also be mapped on the general model of a coupled system. The RNOs can be mapped to the symbolic processing unit, the selection rules part of the PNOs and communication manager to the coupler and the remaining part of PNOs can be seen as the numerical processing units.

15.4.2. Example: Coupled System in Computer Vision

In this section we illustrate the concept of coupling with an example from image understanding for two primary reasons:

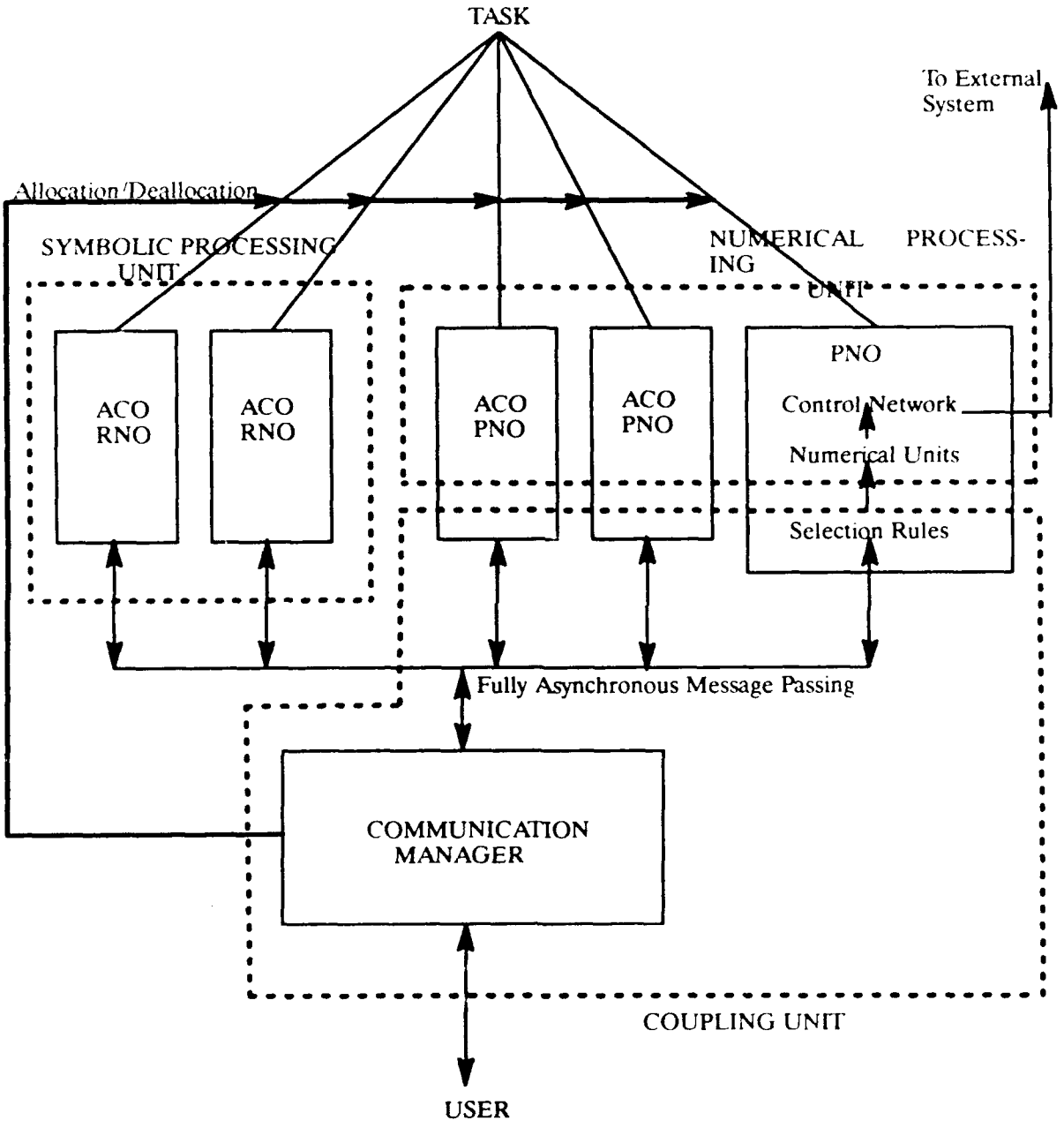


FIGURE 8. Model of Coupling Autonomous Comm. Objects

1) Image understanding is a fairly important research topic in its own right with several aspects of both theoretical and practical interest.

2) A typical image understanding system is a fairly complex knowledge-based system where the coupling process occurs at several stages of problem solving. The interaction between the various stages of problem solving is not yet well understood.

We illustrate the applicability of the model by choosing an image understanding example, namely 3-dimensional object recognition. The coupled system has hybrid coupling strategy, distributed architecture and various stages of coupling.

15.4.2.1. Scope of the Problem

The problem that we have chosen to illustrate our idea is that of 3-D object recognition. This problem basically involves the recognition of a 3-D object from its 2-D projection in a scene. In a typical scene that we consider, there are instances of multiple objects (i.e. objects of different types) and also multiple instances of a single object. The objects that we have considered are polyhedral (i.e. bound by planar surfaces). Each object is described in its local coordinate reference frame in terms of a 3-D wire-frame model. The recognition process essentially involves computing of coordinates, that would cause the 3-D wire-frame model of the object to project onto the 2-D scene, as actually observed. The coordinate-frame transformation is characterized by six parameters namely three degrees of translation, one along each coordinate axis and three degrees of rotation, one about each coordinate axis. Solving for these six parameters is what is commonly referred to as camera viewpoint determination. The block diagram of typical 3-D object recognition system is shown in Figure 9.

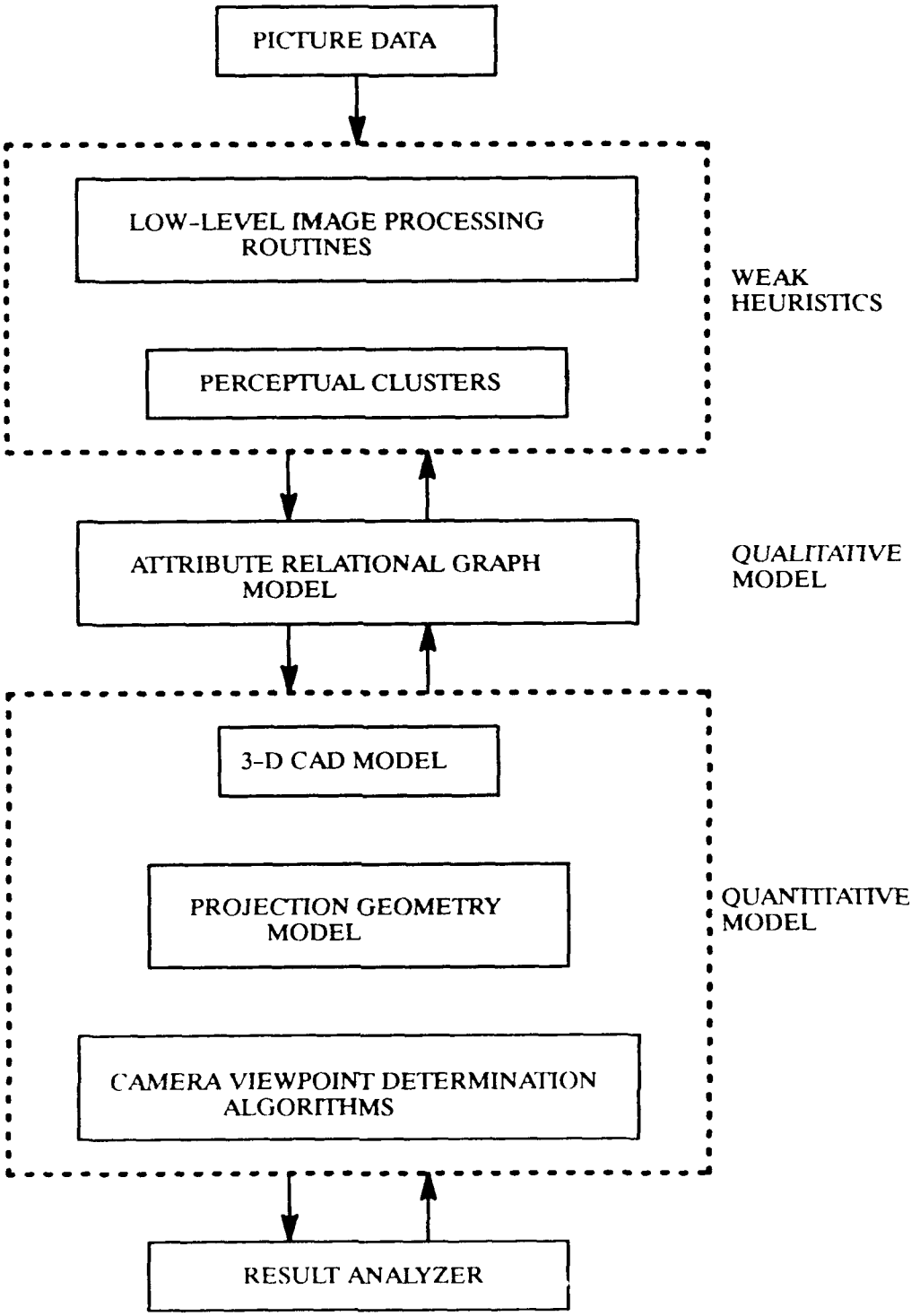


FIGURE 9. Block Diagram of A 3-D Object Recognition System

15.4.2.2. Coupling of Weak Heuristics with a Qualitative Model

For the specific problem that we have considered, the data-directed weak heuristics would mainly consist of perceptual clusters in the form of junctions, parallel lines and collinear lines. Perceptual clustering of image primitives has evoked a fair deal of interest in the image understanding community recently. Lowe [18] and Walters [19] have noted that perceptual clustering of visual cues is an important aspect of low level vision. Perceptual clustering techniques are weak because they have no a priori knowledge about the objects in the scene but rely mainly on locally observable patterns in the image. Perceptual clustering processes are included in the NPU.

The qualitative model is supposed to capture in some sense a certain qualitative understanding of the detailed quantitative model under consideration. In our example the qualitative model of the object is represented by an attribute-relation graph which describes node-adjacencies and instances of parallel lines and collinear lines. This qualitative model is a qualitative description of the 3-dimensional topology of the object. The qualitative model is stored in the knowledge base in the SPU.

The coupling process in this case is basically an instantiation of the nodes and arcs of the model graph of the object with instances of certain perceptual clusters in the scene. The perceptual clusters belong to a single object from a projection graph. The NPU provides the result of perceptual clustering process to the SPU, which is represented by labels. In the SPU the instantiation process amounts to detecting a subgraph in the model graph that is isomorphic to the projected graph (subgraph isomorphism).

At various stages during the clustering process, there exist several choices regarding how to group perceptual clusters. Our chief objective is to ensure that clusters within a single group should have arisen from the projection of a single object. Such an unambigu-

ous grouping of perceptual clusters is not possible without using some knowledge about the object in question. It is therefore necessary at some stage to use the qualitative model to guide the grouping process. Using the qualitative model to guide the grouping process, namely model-driven clustering, has two clear advantages in terms of

1) Resolution of ambiguities during the grouping process
2) Pruning the number of groupings to be considered.

During the process, the CU translates symbolic information from SPU to low-level specification for the NPU and invokes numerical processes necessary to perform perceptual clustering guided by the qualitative model.

15.4.2.3. Coupling of a Qualitative Model with a Quantitative Model

The invocation of the appropriate quantitative model in NPU involves:

(1) Instantiating the 3-dimensional CAD model of the object.
(2) Invocation of the appropriate model for projection geometry, i.e. perspective, affine, orthographic, etc.
(3) Choice of appropriate algorithms for camera viewpoint determination.

The qualitative model reflects the 3-D topology of the object in terms of collinear lines, parallel lines and junctions. These are the qualitative aspects of the quantitative model which are invariant over a large range of viewing orientations and subsequent projection. Other aspects of the quantitative model, such as lengths of lines, angles between lines, etc., are not represented in the qualitative model since they vary greatly with viewing orientation and, hence, are of little use in the first phase of coupling. The qualitative model in the SPU should thus represent just enough knowledge about the quantitative model so that quan-

titative model is appropriately and unambiguously invoked by the SPU, when the situation arises. In this case the CU activates the selected numerical processes which reflect the invoked quantitative model. In our problem the computation of view point parameters represents the precise recognition of an object in a scene. An instance of an object is said to be present in a scene if and only if there exists a set of viewpoint parameters that project the 3-D model of the object to match the scene data to a given degree of accuracy.

15.4.2.4. Coupling of the Quantitative Model with the Result Analyzer

The result analyzer in the SPU will get the computed result from the NPU and verifies the view point parameters by checking whether these parameters lie within the prespecified bounds and whether the projected 3-D model matches the image data within reasonable bounds. If computed viewpoint parameters fail to give a reasonable match then the result analyzer has to reason about the cause of failure and suggests suitable alternatives. Typically the result analyzer would have to take into consideration the following factors:

(1) If the algorithm for viewpoint determination is an iterative one, such as Newton-Ralphson, then convergence properties, initial viewpoint estimates and possibility of the solution being trapped in a local minima have to be considered.

(2) Selection of an alternative solution to the subgraph isomorphism determination process in the first phase of the problem solving process.

(3) Check for the presence of occluding objects in the vicinity of the objects being verified. Occlusion could cause degradation in the quality of the match.

The final phase of the problem-solving process is thus a very crucial one. The problem-solving strategy depends heavily on the analysis done by the result analyzer.

15.4.2.5. Hybrid Control Strategy for Coupling

In our problem of 3-D object recognition, the goal of the problem-solving process should be unambiguous recognition of objects in the scene. Our weak heuristics in the form of perceptual clusters do not guarantee that in any way. There is no way one can fully determine whether a certain cluster or a group of clusters has arisen from a particular object unless one takes recourse to some sort of knowledge about the models present in memory.

We propose a coupling scheme based on evidential reasoning for coupling weak heuristics with a qualitative model. The scheme could be briefly described as follows:

(1) Each cluster is assigned a measure of belief based on the premise that it has not arisen by accident.

(2) Each group of clusters is assigned a belief measure based on the belief measures of the participating clusters and the premise that these clusters belong to a single object (without knowledge about the exact object).

Both (1) and (2) necessitate the formulation of a belief measure and the formulation of an appropriate combining function [20][21]. This phase of the clustering process is purely data-directed and is carried out entirely in the absence of any knowledge about the objects present in memory.

Every cluster generated in the data-directed phase is simultaneously evaluated by models based on

1) The ability of the cluster to constrain the qualitative model,

2) The discriminating ability of the cluster.

The model-centered evaluation can be expressed in the form of a goodness measure such as max(quality of match)/sum(quality of match). Thus each cluster has an effective belief measure which is the combination of the data-directed belief measure and the model-centered belief measure. As soon as the belief measure exceeds a prespecified threshold, the qualitative model in question guides the clustering process. This brings about a transition from the use of data-directed heuristics to model-directed heuristics. This sort of coupling scheme naturally implies an object-centered distribution of the qualitative models wherein each model is treated as an independent agent. Such a distribution encourages the model-centered belief evaluation process to be carried out in parallel by the models in question.

15.4.2.6. Distributed Architecture

The clustering heuristics could be distributed as heuristics for parallel lines, heuristics for junctions and heuristics for collinear lines. Such a distribution would be a functional distribution. The strategies for distribution could be *either cooperative or competitive*. For example, in the clustering shown in Figure 10, L1, L2 and L3 form a junction whereas L2 and L4 constitute parallel lines. If the clustering process were competitive then the arbitration could be done on the following basis:

(1) L1 and L3 forms a junction cluster whereas L2 and L4 form a cluster of parallel lines.
(2) L1, L2 and L3 form a junction cluster and L4 is not assigned to any cluster.

If the clustering process were cooperative then L1, L2, L3 and L4 would form a complex cluster. Such a complex cluster would be desirable since it would add greater constraint than either cluster having been considered independently.

There exist iterative algorithms for the determination of camera viewpoint parameters. Typical examples are Newton-Ralphson and Simulated Annealing. Both techniques

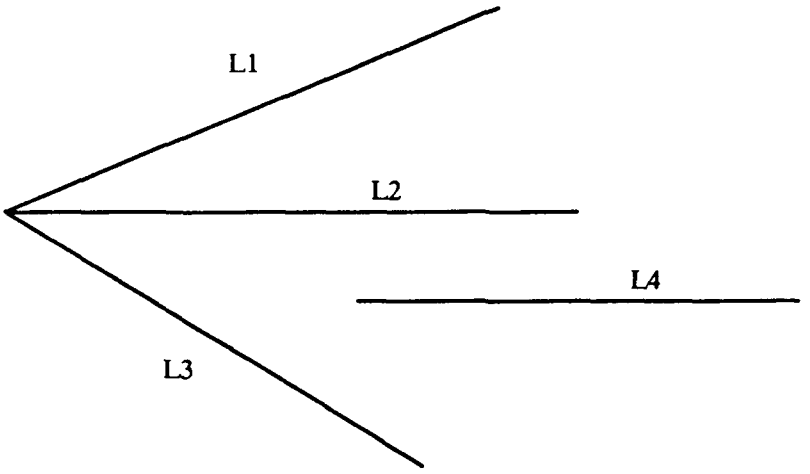


FIGURE 10. A Typical Clustering Example

are based on the principle of minimizing the sum of squared error between the projected model and the image data. Newton–Ralphson converges quickly in the vicinity of the solution but also tends to get trapped in local minima. Simulated Annealing is slow but robust. If there is doubt about the initial estimates for the viewpoint parameters then it is wiser to use Simulated Annealing until one is in the vicinity of the absolute minima and then pass the results on to the Newton–Ralphson routine. In most cases a cooperative mode of coupling is preferable to a competitive mode since in a cooperative mode more effective use is made of the available resources.

In a distributed architecture, communication between agents is very important to perform a cooperative mode operation. The CU has to deal with complex issues such as:

1) nature of interactions between agents

2) the means of communication between agents

3) stability of the system and guaranteed convergence to a solution.

KIM, as explained in the section 15.4.1.3. is a good example of the CU dealing with these complex issues.

15.4.3. Implementation Example in O.O.P. Paradigm

3-D MOSAIC by Herman and Kanade [22] is a well-known computer vision system which extracts symbolic descriptions from the multiple view of urbane scenes. It has many symbolic and numerical procedures, and can be constructed as a coupling system. Since it is a large system, only a portion of the system is used for our coupling example: only the scene model handling part. Figure 11 shows a flow chart of 3-D MOSAIC.

The scene model handling part can be further divided into 4 levels. The lowest level is the actual data description. 3-D Mosaic has a unique scene model called the "structure graph". It is a symbolic description of the scene model and can be updated as needed. It has a graph structure which has part-of and constraint links and its nodes can be dots, line, planes, vertices, edges and faces. Dots, lines and planes are geometrical primitives and vertices edges and faces are topological primitives.

The next level is the constraint handler. It propagates constraints throughout the graph to make it consistent. Constraint can be introduced by adding/deleting nodes or links. Changes in nodes and links can generate geometrical or topological constraints, or both. There are several rules for constraint propagation. This level should have a rule-base and some method with which to apply it.

The upper level is for merging structure graphs. It merges partially completed structure graphs or raw data from lower processes into one object. It has its own rules to solve inconsistency and ambiguities.

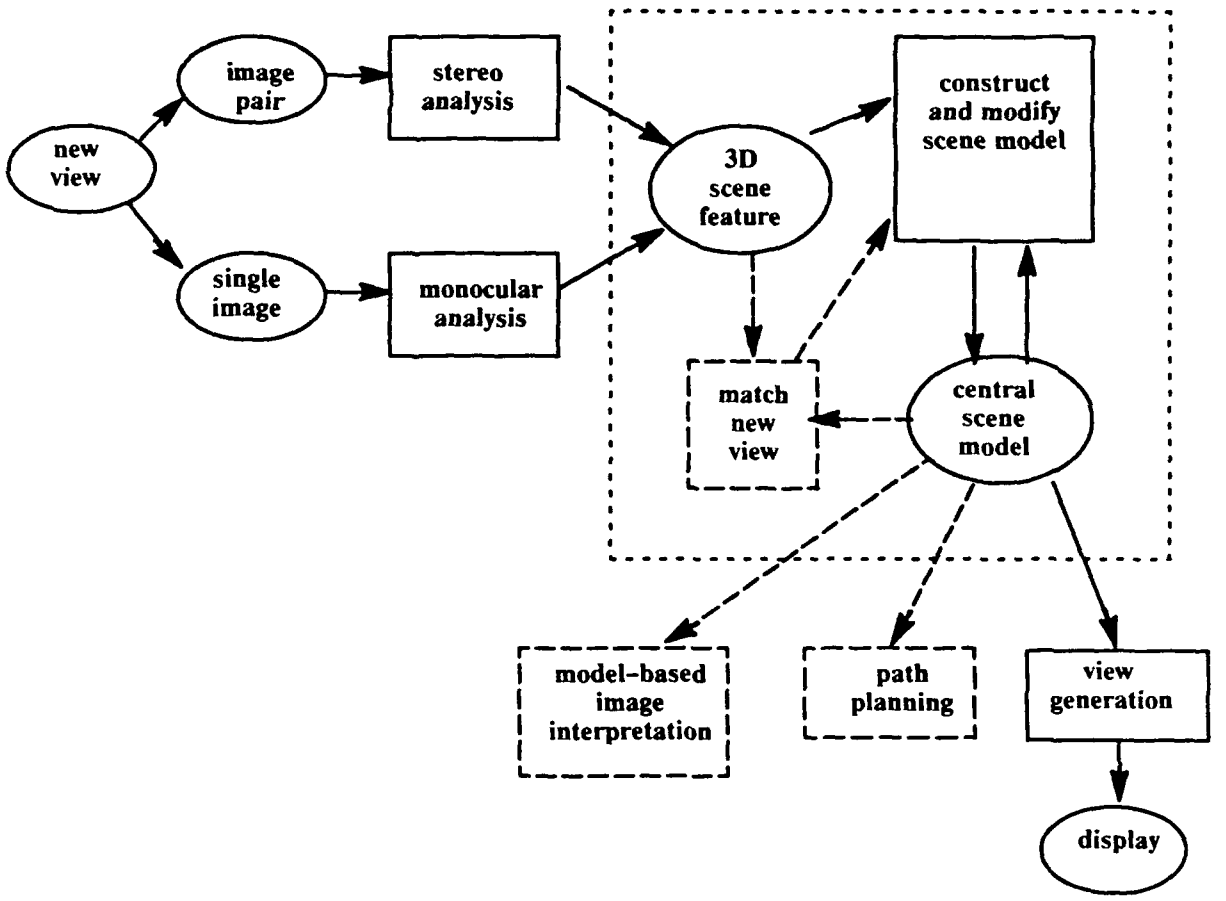


FIGURE 11. 3-D Mosaic Flowchart

The fourth level is for combining new views. It combines a new view to an existing scene model. The following table shows these levels and their symbolic and numerical procedures.

Level	Symbolic processes	Numerical processes
Structure Graphs	. Add & delete links . Add & delete nodes	
Constraints Handler	. Rules of constraints propagation (geometric and topological) . Redundancy removal	. Equations for dots lines and planes . Inclusion relations between them
Merging Structure graphs	. Rules from knowledge of planar-faced objects and urbane scenes. . Hypothesize new edges and vertices	. Weighted average of two edges . Intersection points . Least-square fittings of planes . Distances between objects . Detecting parallelism between lines
Combining new view	. Matching . Rules for discrepancies . Rules for merging	. Scale and coordinate transformations . Inclusion relations . weighted average . detecting Parallelism

Table. Symbolic and numerical processes in Model Handling

When these procedures are implemented in an object-oriented paradigm, each level can be a big object and numerical and symbolic procedures can be treated as objects which receive messages from other objects. One advantage of using the object-oriented approach is that its modules can be used elsewhere because their interfaces are well-defined. In this example some numerical procedures can be shared with different levels of objects and they

also can be used in lower level processing like stereo analysis. Couplings between symbolic and numerical procedures are accomplished in the form of messages. This gives a great freedom to programmers. Since the interface between modules is well-defined, a programmer does not need to wait for modules to finish their tasks before sending more messages. This also enhances the quality of the modules.

15.5. NEED FOR COUPLING IN VISION SYSTEMS FOR 3-D SCENE INTERPRETATION

Owing to the vast amount of literature focused on three-dimensional scene interpretation, it would not be possible to enumerate every existing three-dimensional scene interpretation system. A categorization of existing three-dimensional scene interpretation systems based on (a) Choice of representation and (b) Choice of control, would be far more illuminating.

15.5.1. Choice of Representation

The issue of representation deals with (i) Choice of image features and (ii) Choice of model features. The choice of image features is often tuned to the nature of the application domain and nature of the sensor used. The choice of representation for the object models is also dependent on the application domain. For the purpose of matching, the representation of the object models is closely tied to the representation chosen for the image features.

The choice of object model representation is based on (i) Surface descriptions or (ii) Volumetric descriptions. Surface descriptions are used extensively in constrained environments where it is possible to extract surface features from active range sensors, stereo imaging or photometric stereo. Volumetric descriptions based on generalized cylinders were first used by Brooks in ACRONYM[23].

Both surface and volumetric descriptions could be either local or global. Local surface descriptions are typically based on local curvature properties of the surface. Besl

and Jain[24], Fan et. al.[25], Brady and Ponce[26] and Vemuri and Aggarwal[27] have described segmentation schemes based on surface curvature properties. Faugeras and Herbert[28] describe a surface segmentation and surface matching scheme based on segmentation of the surface into planar and quadric surface patches. Global surface descriptors are based on describing the surface in terms of a few global parameters such as super-quadrics. Bajcsy and Solina[29] describe an iterative surface fitting technique for extracting the global shape parameters for super-quadrics. The surface normal distribution such as the Extended Gaussian Image (EGI) which is a global surface descriptor has also been used by a few researchers[30]. Generalized cylinders are examples of global volumetric descriptors whereas voxel-based oct-tree representations[31] are local volumetric descriptions. Local features are well suited for recognition in the presence of occlusion whereas global features enable rapid recognition in those situations where the image data is unoccluded.

Features could also be classified as generic or distinctive. Generic features cover a broad range of objects under the domain of interest. Distinctive features on the other hand make specific assumptions about the objects in question such as in the 3-DPO vision system[32]. Distinctive features sacrifice generality for recognition speed whereas generic features sacrifice recognition speed for generality.

15.5.2. Choice of Control Strategy

A scene interpretation hypothesis can be looked upon as a solution to a constraint satisfaction problem. The approach for arriving at a consistent scene interpretation is therefore a constraint-directed search through the space of possible scene interpretations. 3-D object recognition systems can be classified according to the search strategy used :

(1) *Top-down or model-driven search* is commonly used in the recognition via localization technique for object recognition. The features used are primitive and matching takes place

very early in the recognition process. The primitive geometric features are matched against similar features in the object model. Matches are checked for local consistency using simple geometric constraints such as distance and angle measurements. A set of locally consistent matches is used to compute a global transformation from the model coordinate system to the scene coordinate system. The *Interpretation Tree* (IT) approach[33] matches a scene feature with a model feature at every stage in the recognition/localization process as shown in Figure 12. Locally consistent matches are represented by a path in the IT. Local geometric constraints are used to prune paths in the IT thereby restricting the possible interpretations. When a path of sufficient length is found, a global transformation is computed. The control structure of the algorithm is that of sequential hypothesize-and-test with backtracking. The *Hough (pose) clustering* approach[34] matches each scene feature to each possible model feature. The matches are constrained by local geometric constraints based on angle and distance measurements. Each match enables one to compute a point in the Hough (parameter) space. Clustering of points in the Hough space yields a globally consistent hypothesis regarding the pose of the object (Figure 13). Model-driven approaches are useful when one is faced with noisy or occluded data which precludes a higher-level abstraction of the data.

(2) *Bottom-up or data-driven driven search* seek to obtain a higher level semantic description. Matching takes place fairly late in the recognition stage. A high-level scene description is matched against a high-level semantic description of the object. The recognition phase is typically based on paradigms such as graph-matching, subgraph isomorphism and maximal clique detection. Recognition is followed by localization for which highly object-specific shape descriptors are used. The advantage of such an approach is that since a higher level description of the data is used, the number of hypotheses generated in the matching phase are fewer in number. The disadvantage is that since this approach requires extensive

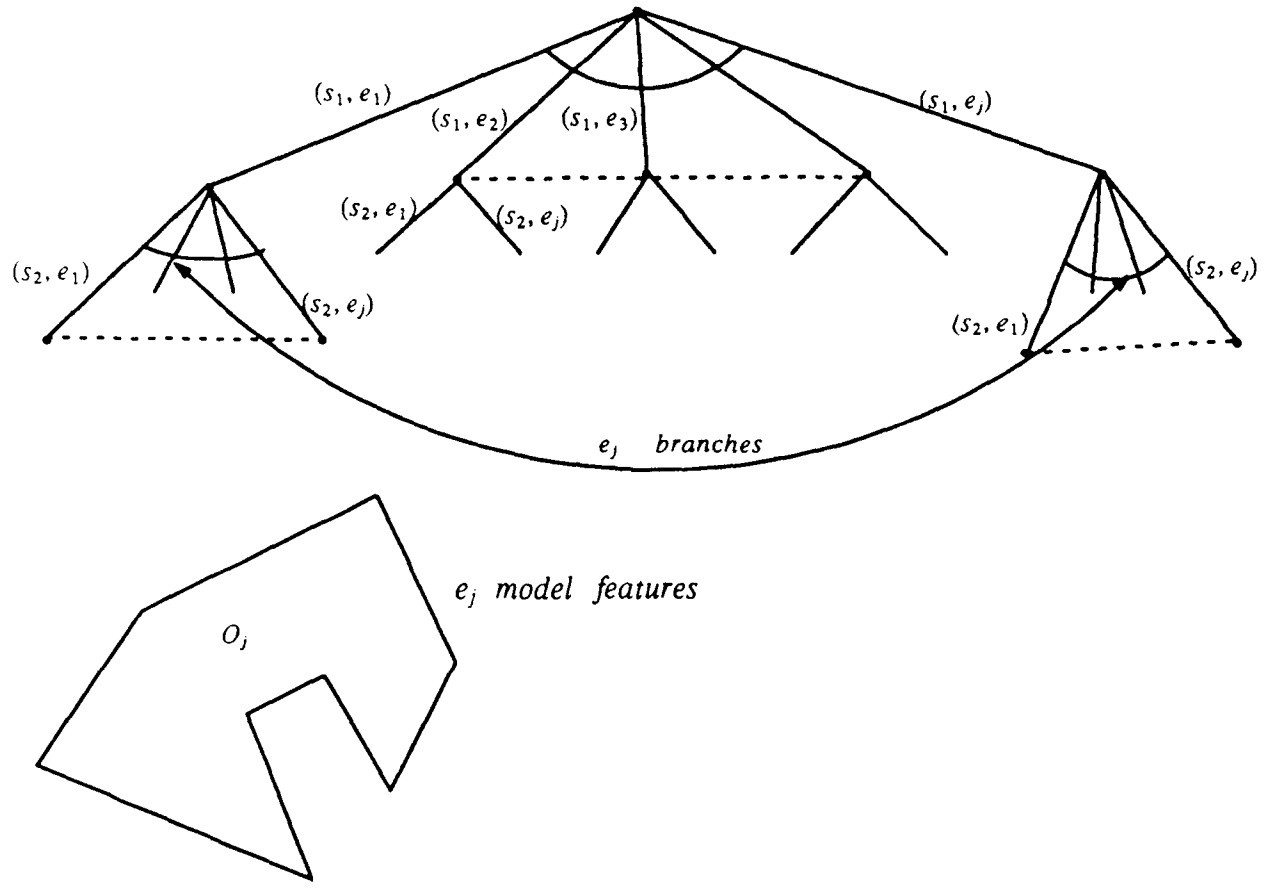


FIGURE 12. Searching through an Interpetation Tree

preprocessing of image data, it requires that the quality of the data be good, i.e. unoccluded and unambiguous.

(3) *Interpretive search strategy* uses the object model interpretively, that is, the knowledge is extracted from the model and transformed into an execution strategy at run time. The system relies on a generic reasoning mechanism such as numerical optimization of some matching criterion, constraint satisfaction by symbolic reasoning, or tree search by hypo-

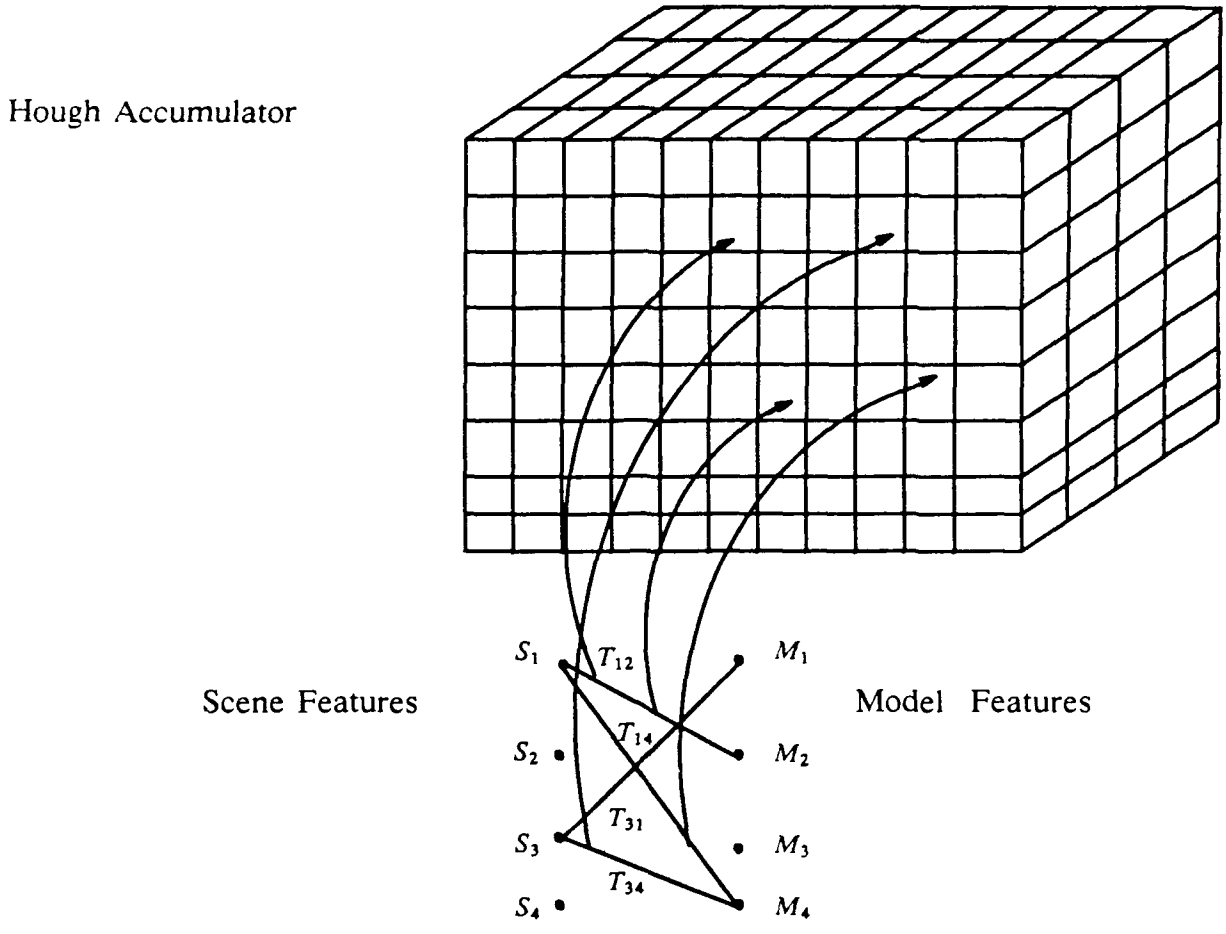


FIGURE 13. Hough (Pose) Clustering

thesize-and-test. The system may not be the most efficient for the particular object in the scene. An interpretive system sacrifices speed for generality and flexibility. ACRONYM is an example of an interpretive system.

(4) *Precompiled search strategy* embeds the relevant control knowledge into the object model and compiles it into a recognition strategy off-line. As a result, little computation is done during the process of recognition. The work by Goad[35] and Ikeuchi[36] are examples of

this approach. The advantage of precompilation is that recognition is fast but at the cost of loss of generality.

It is possible to further categorize search strategies into *distributed vs. centralized* search strategies. In distributed search several processes cooperate to come up with an interpretation whereas in centralized search the control strategy is located within a centralized control module. Most of the 3-D object recognition systems to date employ a centralized search strategy.

15.5.3. Shortcomings of 3-D Scene Interpretation Systems

Our study of the state-of-the-art brings out certain fundamental shortcomings of existing computer vision systems especially those that deal with three dimensional scene interpretation. Of these, the two most important ones are : *(i) Inadequacy of any one single level of representation* and *(ii) Inadequacy of a constraint propagation technique based on any one single level of representation*. There are two broad categories of representation: (i) Graph-theoretic representations which typically use global features and (ii) Representations which use local, primitive geometric features. Graph-theoretic constraint propagation and constraint satisfaction techniques such as subgraph isomorphism and maximal clique detection are well suited for graph-theoretic representations. On the other hand, for representations which use local, primitive geometric features, constraint propagation/constraint satisfaction techniques that rely on propagation of geometric constraints that arise out of the matching of local geometric features are used. Both have their advantages and disadvantages. Graph-theoretic representation and constraint propagation techniques based on global features enable fast and efficient recognition but are not robust to occlusion. Representations based

on local geometric features are robust to occlusion but the corresponding constraint satisfaction techniques are computationally intensive. State-of-the-art 3-D scene interpretation systems are committed to one or the other level of representation and constraint propagation. For both robustness and efficiency in 3-D scene interpretation, both levels of representation and constraint propagation need to be effectively integrated into a single framework. Since graph-theoretic representations require extensive preprocessing of the image data, the control strategy could be said to be largely *data-driven*. Recognition and localization techniques which rely on representation, and constraint propagation techniques based on primitive geometric features, have a *model-driven* control strategy since the preprocessing of image data is minimal and matching takes place very early in the recognition and localization process. State-of-the-art 3-D scene interpretation systems, due to commitment to one or the other level of representation or constraint propagation, are also committed to one or the other form of control strategy. Thus, integration of both levels of representation and constraint propagation also entails integration of both model-driven and data-driven control strategies.

Another shortcoming of existing 3-D scene interpretation systems is *the lack of effective integration of symbolic and numerical processing*. Feature extraction and constraint propagation techniques which operate on low-level image representations deal with largely numerical data whereas constraint propagation techniques which deal with high-level image representations deal with symbolic data. Thus, an effective integration of multiple levels of representation and constraint satisfaction also entails an effective integration of symbolic and numerical processing.

Another shortcoming of existing 3-D scene interpretation systems is the fact that *the scope for parallelism has not been fully exploited*. Low-level processes such as edge extraction, surface fitting, and region growing are easily amenable to parallelism and have been well-

discussed in the literature. On the other hand, parallelization of high-level vision processes has received much less attention and needs to be explored.

A *coupled systems* approach to problem-solving in 3-D scene interpretation and computer vision in general would alleviate most of the shortcomings of existing 3-D vision systems. We propose the use of *object-oriented* techniques for integrating symbolic and numerical processing in computer vision. Object-oriented programming techniques have several features of interest which can be exploited in the context of computer vision:

(1) *Representation at multiple levels of granularity :* By use of object-oriented techniques for representation, objects can be represented at multiple levels of granularity. Organization of the representation into PART/SUBPART and CLASS/SUBCLASS hierarchies makes the choice of appropriate granularity of representation possible.

(2) *The constraint propagation technique is made to be part of the representation* : This enables the problem-solving mechanism to choose the constraint propagation mechanism best suited for a particular granularity of representation. Constraint propagation and constraint satisfaction can be achieved at multiple levels of abstraction. Representation of both geometric primitives and geometric constraints as objects makes this possible.

(3) *Flexibility of control structure* : Since the constraint propagation/constraint satisfaction mechanism is a part of the representation, aspects of control which are specific to a particular representation are encapsulated within the object definition. There are two aspects of control: (i) *Strict top-down planning*, in which the sequence of operations are known a-priori. This is typical of a precompiled control strategy; And (ii) *Searching*, in which the sequence of operations are obtained by searching through a

search tree of alternatives using a hypothesize-and-test procedure. This is typical of an interpretive control strategy. One could envisage a control strategy in which the overall control strategy is organized as a search through precompiled plans. By treating precompiled plans as components of an object-oriented representation, one can achieve both speed of precompilation and the flexibility of an interpretive control scheme.

15.5.4. Implementation Issues

Practical implementation of an object-oriented system for coupling symbolic and numerical processes in 3-D scene interpretation systems would involve (a) choice of objects, (b) choice of descriptors which would form the indexing criteria for the objects, (c) choice of methods to be associated with each object, and (d) design of control schemes suited for an object-oriented environment

Choice of Objects :

The object types that we have in mind are : (1) Primitive objects, (2) Composite objects, (3) Constraint objects, (4) Class objects, and (5) Matched objects.

(1) Primitive objects : As pointed out by Ramamoorthy and Sheu[37], the choice of primitive objects and the choice of representation granularity for these primitive objects is crucial to the design of any object-oriented system. A choice of representation that is very coarse-grained may lead to loss of detail whereas very fine-grained representation for objects may lead to excessive overhead in inter-object communication. For our specific problem, we have a natural hierarchy of primitive objects. These are: (a) points, (b) boundary segments which are a collection of adjacent boundary points and which satisfy a certain smoothness or continuity criteria, (c)

boundaries which are a collection of boundary segments linked together (Boundary segments are linked together at points of discontinuity, i.e. corners or cusps), (d) surface regions which are a collection of surface points, and (e) surface patches which are made up of surface regions bound by surface boundaries. For our purpose, we will initially choose surface regions and surface boundary segments as the primitive objects for the purpose of recognition. Points will be treated as implicit components of the representation for surface regions and surface boundary segments rather than as independent objects.

(2) Composite objects : Composite objects will be created from primitive objects. The process of composition creates a PART-OF hierarchy. Composite objects contain a list of their component parts and constraints between their component parts. Creation of composite objects is necessary for: (a) data abstraction, i.e. generating a higher level description of scene data, and (b) representing complex object models in terms of their simpler constituent component parts.

(3) Constraint objects : Constraints between objects will be represented explicitly as objects. Each constraint object contains slots for: (a) The constraint type, (b) The list of object types that qualify for the constraint, and (c) Symbolic/Numerical representation of the constraint.

(4) Class objects : All of the previously mentioned objects could be arranged in a CLASS-SUBCLASS hierarchy. A typical class object contains slots for: (a) Object type, (b) Constraints on the object type which provide the class definition, and (c) Slots for class specialization/generalization. Class objects will be used to classify primitive objects into object classes. The approach that we will pursue for the classification of primitive objects is as follows: For surface regions the classification is done

based on the signs of the Mean and Gaussian curvature. The Mean and Gaussian curvature values are used to classify surfaces into eight basic qualitative surface region types. This classification is further refined based on metrical values of the Mean, Gaussian and principal curvature values. For surface boundary segments, the classification is done based on qualitative attributes such as boundary edge type, i.e. step, roof or crease, and boundary geometry, i.e. straight, curved, closed, etc. These qualitative attributes are further refined based on metrical properties of curvature. Object models are classified based on: (i) number and orientation of subparts, and (ii) dimension of the object models and/or dimension of their subparts etc. Constraints are classified either relational or transformational. Relational constraints constrain the relative geometry of two objects whereas transformational constraints compute the relative coordinate transformation between two objects. Relational constraints are classified based on qualitative properties such as adjacency and containment. These constraints are further refined based on distance between centroids, relative orientations based on curvature values, length of the common boundary, etc. These classifications need not be a strict tree-based hierarchy in many cases. A lattice-based hierarchy with multiple inheritance would be allowed.

(5) Matched objects : Matches between objects generated from the scene description and prestored objects will be represented as objects. The matched objects contain slots for: (a) describing the objects matched, and (b) constraints imposed by the matching process. The constraints imposed by the matching process serve two purposes: (i) to refine the classification of the objects by propagation of constraints through the CLASS-SUBCLASS hierarchy, and (ii) to determine or refine the pose of the object with respect to the viewer.

Choice of Descriptors for Objects :

The descriptors for objects will be used to form the indexing criteria by which the objects should be accessed. These descriptors will be qualitative at coarse levels of description. For surface regions the signs of the Mean and Gaussian curvature may serve as qualitative indexing criteria whereas for boundary segments qualitative attributes such as step, roof, straight and curved may serve as qualitative indexing criteria. These qualitative descriptors will be further refined to more quantitative or metrical descriptions based on curvature properties.

Representation of Methods associated with each Object :

In object-oriented approaches the numerical procedures (or methods) should be encapsulated along with the symbolic description of the objects they act upon. For our problem the important procedures to be represented are: (i) low-level algorithms such as segmentation and feature extraction, and (ii) control procedures such as constraint propagation and precompiled recognition strategies.

In object-oriented approaches the numerical procedures (or methods) should be encapsulated along with the symbolic description of the objects they act upon. For our problem the important procedures to be represented are: (i) low-level algorithms such as segmentation and feature extraction, and (ii) control procedures such as constraint propagation and precompiled recognition strategies.

The methods associated with each object are :

(1) Algorithms for image segmentation associated with a certain object type : Most image segmentation algorithms are fine-tuned for a certain object type. For example detection of step edges needs a different set of operations as compared to detection

of roof or crease edges. Segmentation algorithms can be also be refined based on the object type. Segmentation algorithms for general surfaces incorporate weak criteria based on qualitative properties for clustering such as continuity, adjacency, and containment. Segmentation algorithms for specific surfaces do incorporate clustering criteria based on very specific assumptions about the surface type. For example, clustering of surfaces based on assumption of cylindrical surfaces would incorporate the assumption of zero Gaussian curvature.

(2) Algorithms for feature extraction associated with a certain object type : These feature extraction algorithms are tuned for a certain granularity of representation of that particular object type. A point-wise surface normal and surface curvature representation is the most general representation for the object-type *surface*. Thus feature extraction algorithms for extraction of point-wise surface curvatures and surface normals are encapsulated in the object-type *surface*. For cylindrical surfaces, a representation based on the length and direction of axis and radius of curvature would be more appropriate. Likewise, feature extraction algorithms for the extraction of the magnitude and direction of the axis is made part of the object *cylindrical-surface*.

(3) Algorithms for constraint propagation and constraint satisfaction suited for a certain granularity of representation of that object type : Constraint propagation and constraint satisfaction techniques for matching are well suited for a certain granularity of representation. Hough clustering techniques are well suited for fine-grained representation whereas graph-theoretic techniques such as sub-graph isomorphism and maximal clique finding are better suited for symbolic representations. These constraint propagation and constraint satisfaction techniques should be made a part of the representation of the appropriate object.

(4) Precompiled recognition strategies : Precompiling recognition strategies for object recognition in a bin-picking scenario has already been addressed by Goad[35] and Ikeuchi[36]. The bin-picking scenario is a rather simplified one compared to the one we are addressing. The bin-picking problem is simplified by the fact that one is only interested in the topmost object in a *pile of objects* and the *identity* of the object is known a priori. In a general object-recognition scenario such as ours, precompiled recognition strategies would have to be triggered when there is sufficient evidence available for the presence of a certain object. These precompiled recognition strategies would be a part of the representation of object model in question along with the criteria for triggering them.

Design of Control Schemes suited for an Object-Oriented Environment : The following issues have to be dealt with in the design of control schemes :

Embedding Semantics : Object-oriented techniques, though well suited for representation, are weak and incomplete when it comes to making inferences. Object-oriented representations need to be augmented by semantics which would form a part of the control structure. The incorporation of semantics (such as rule-based semantics) should greatly enhance object-oriented representation by saving the need for explicit representation in many cases. The following type of semantics will be embedded in our framework:

(1) Semantics for method specialization : Since the objects are arranged along several hierarchies such as CLASS-SUBCLASS and PART-SUBPART it is necessary to come up with semantics for method/specialization and method generalization. By the use of semantics one should be able to generalize/specialize methods by traversal of the object hierarchy.

(2) Semantics for multiple inheritance : As mentioned earlier, it is possible for objects to inherit properties from more than one class. Thus it is necessary to develop semantics for method specialization/generalization which would take into account multiple inheritance.

(3) Design of message passing protocols between objects : Design of message passing between objects should take into account: (i) method selection, (ii) argument passing to the appropriate method, and (iii) propagation of the result.

How this semantic knowledge is to be represented is a difficult problem. Our first attempt at this problem will be to embed the semantics for specialization in the most general object category possible. The problem of multiple inheritance is more difficult and requires a careful study. Our aim is to keep the control as simple as possible. Our policy would be to encapsulate much of the semantics which is specific to a particular object type along with the definition of that particular object.

Control of problem-solving in multi-layered constraint nets : At any stage in the recognition process, the state of problem-solving will be expressed as a network of constraints. The nodes in the network correspond to objects, and links in the network correspond to constraints between the objects. Both objects and constraints between objects will be expressed at multiple levels of resolution and multiple levels of refinement along CLASS-SUBCLASS and PART-SUBPART hierarchies. Control of problem solving in multi-layered constraint nets is therefore an important issue that will have to be studied.

In order to keep the overall control of problem solving as simple as possible, we plan to organize the top-level control as a simple search process through multiple hierarchies coupled with Waltz filtering to improve efficiency. Control strategies which are

specific to a certain object definition are made a part of the representation of that object. Since the descriptions of objects and constraints at a coarse level of resolution are qualitative, qualitative reasoning will play an important role in pruning the space of possible interpretations. The search process through multiple hierarchies could be organized as :

(1) Data-driven Search : Searching upwards through the PART-OF hierarchies for combining hypotheses. This is an essential part of data abstraction.

(2) Model-driven Search : Searching downwards through the PART-OF hierarchies to search for missing objects. Feature extraction and segmentation techniques associated with those objects are invoked during the downward search. This is an essential aspect of model-guided segmentation.

(3) Goal-directed Search : Searching downwards through the CLASS-SUBCLASS hierarchy in order to refine the classification.

(4) Failure-directed search : Assuming that objects in the same level of the hierarchy are connected by *similarity* links, a sideways traversal across the *similarity* links in the event of failure would constitute a failure-driven search.

All the above-mentioned issues need to be addressed in the practical implementation.

15.6. STATUS OF CURRENT IMPLEMENTATION

In order to study coupled systems in the context of computer vision, the problem of 3-D object recognition from range data in a multiple-object scene with partial occlusion was considered. This problem is of considerable theoretical and practical interest. It is encountered in several scenarios such as robot bin-picking, automated industrial inspection, autonomous navigation, etc. The two main problems encountered when dealing with a multiple-object scene are: (i) combinatorial explosion of the search space of scene interpretations, and (ii) generation of spurious scene interpretations. Thus, the issues of *representation* and *constraint propagation/satisfaction* were dealt with primarily with the following objectives in mind: (a) reducing the combinatorial complexity of the search space of possible scene interpretations, and (b) ensuring robustness against occlusion.

Our work thus far has brought out the advantages of using qualitative features to achieve these objectives. With Hough clustering as the chosen constraint propagation/satisfaction technique, three problem scenarios of increasing complexity were used to demonstrate the effectiveness of using qualitative features for recognition and localization: (i) recognition of polyhedral objects, (ii) recognition of curved surfaces, in particular, conical, cylindrical and spherical surfaces, and (iii) complex objects made up of piecewise combinations of conical, cylindrical, spherical and planar surfaces. The results are analyzed both in terms of accuracy and robustness against occlusion. The choice of Hough clustering as the constraint propagation/ constraint satisfaction technique was governed by the fact that it is easily amenable to parallelism.

This research clearly shows the role of qualitative features in recognition and localization. Qualitative features provide :

(1) An effective means of reducing the combinatorial complexity of the search space of

possible scene interpretations. In the context of Hough clustering this translates to being able to suppress spurious peaks in the Hough space.

(2) An effective criteria for choosing the appropriate representation for the image data (scene features), object models (model features) and constraints resulting from matching scene features to model features. In the context of Hough clustering this translates to choosing the appropriate parameter space in which to compute the object pose.

The advantage of using qualitative features for recognition and localization was brought out through our experiments in all three aforementioned problem scenarios. The use of qualitative features was shown to greatly enhance the performance of conventional Hough clustering in terms of both: (i) robustness of the recognition process, and (ii) accuracy of the localization process.

The use of qualitative features can be seen as a criteria for selection of the appropriate granularity of representation and constraint propagation. Thus qualitative features would serve as an indexing criteria in a coupled systems approach to problem-solving in computer vision.

15.7. FUTURE DIRECTIONS

The work done so far has brought out the advantages of using qualitative features as a means for indexing into appropriate representations and method selection for three-dimensional object recognition. The fu'ure directions can be briefly outlined as follows :

(1) Implementation of an object-oriented framework for 3-D object recognition in C++ or CLOS.

(2) Tackling other vision problems in the framework of coupled systems namely motion, texture and stereo.

(3) Parallel implementation of constraint propagation / constraint satisfaction algorithms for vision problems.

REFERENCES

[1] M.A. Fishler,
"Parallel Guessing: A Strategy for High Speed Computation",
Pattern Recognition, Vol.20, No.2, 1987, pp 257-263.

[2] Steven D. Campbell and Stephen H. Olson (Lincoln Lab., MIT),
"WX1-an Expert System for Weather Radar Interpretation",
Coupling Symbolic and Numerical Computing in Expert System I,
Elsevier Science Publishers B.V. (North-Holland),
(CSNCES I), 1986, pp 329-348.

[3] Jay C. Ferguson, R.W. Siemens and R.E. Wagner,
"STAR-PLAN: a Satellite Anomaly Resolution and Planning System",
CSNCES I, 1986, pp 273-281.

[4] J. Guillermard and J.C. Lagache,
"PMS-1: A Real-Time Advisor for the Boron Control in PWR
Neuclear Reactor"'
CSNCES I, 1986, pp 284-293.

[5] S.N. Talukdar et al.,
"A System for Distributed Problem Solving",
CSNCES I, 1986, pp 59-67.

[6] T. Sivasankaran and M. Jarke,
"Coupling Expert Systems and Acturial Pricing Models",
CSNCES I, 1986, pp 263-271.

[7] Walt Conley and Uttam Sengupta,
"Intelligent Simulation Models: A Demographic Simulator with
Heuristic Reasoning",
CSNCES II, 1988, pp 173-183.

[8] Dave Thomas,
"What's in an Object",
BYTE Magazine, March,1988, pp 231-240.

[9] Geoffrey A. Pascoe,
"Elements of Object-oriented Programming",
BYTE Magazine, August, 1986, pp 139-144.

[10] Jay W. Tompkins (Amoco Production Co.),
"Using an Object-Oriented Environment to Enable Expert System
to Deal with the Existing Programs", CSNCES I, 1986, pp 161-168.

[11] Christopher A. Powell and Mary J. Cole (A.I. Center, FMC Central Lab.),
"Rule-based Residential Floor Plan Generator",
CSNCES II, 1988, pp 237-248.

[12] "The Complete Guide to MRS",
Tech. Report STAN-CS-85-1080,
Stanford Univ., Computer Science Dept., June, 1985.

[13] David R. Throop (Univ. of Texas, Austin),
"Getting Rules to Talk to Models",
CSNCES II, 1988, pp 259-274.

[14] Jerome Connor and W.M. Kim Roddis (Dept. of Civil Eng., MIT),
"Reasoning for Fatigue and Fracture in Bridge",
CSNCES II, 1988, pp 249-257.

[15] Naseem A. Khan and Ramesh Jain (Michigan Univ.),
"Explaining Uncertainty in a Distributed Expert System",
CSNCES I, 1986, pp 101-113.

[16] Lothar Winkelbauer (Austria),
"A Structure for the Incremental Construction of Coupled Systems",
CSNCES II, 1988, pp 145-155,

[17] Janos Sztidanouvits and Byron Purves,
"Coupling Symbolic and Numeric Computations in a Real Time Distributed Environment.", CSNCES II, 1988, pp 117-128.

[18] D.G. Lowe,
"Three-Dimensional Object Recognition from Single Two-Dimensional Image", Artificial Intelligence, Vol. 31, No. 3, March 1987, pp 355-395.

[19] Deborah Walters,
"Selection of Image Primitives for General Purpose Visual Porcessing", CVGIP 37, 1987, pp 261-298.

[20] H.E. Kyburg,
"Bayesian and Non-Bayesian Evidential Updating",
Artificial Intelligence, Vol. 31, No. 3, March 1987, pp 271-293.

[21] Judea Pearl,
"Distributed Revision of Composite Beliefs",
Artificial Intelligence, Vol. 33, 1987, pp 173-215.

[22] Martin Herman and Takeo Kanade
"The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes for Complex Images",
Readings in Computer Vision, California, Morgan Kaufmann Publishers, 1987, pp 471-482.

[23] R.A. Brooks,
"Symbolic Reasoning Among 3-D Models and 2-D Images",
Artificial Intelligence, 17(1-3), 1981.

[24] P. Besl and R. Jain,
"Segmentation Through Variable Order Surface Fitting",
IEEE Trans. Pattern Analysis and Machine Intelligence, 10(2), March 1988, pp 167-192.

[25] T.J. Fan, G. Medioni and R. Nevatia,
"Segmented Descriptions of 3-D Surfaces",
IEEE Journal on Robotics and Automation, 3(6), Dec. 1987, pp 527-538.

[26] M. Brady and J. Ponce,
"Toward a Surface Primal Sketch",
IEEE Conference on Computer Vision and Pattern Recognition,
1985, pp 420-425.

[27] B.C. Vemuri and J.K. Aggarwal,
"Curvature-based Representation of Objects from Range Data",
Image and Vision Computing, 4(2), May 1986, pp 107-114.

[28] O.D. Faugeras and M. Herbert,
"The Representation, Recognition and Locating of 3-D Objects",
The Int. Journal of Robotics Research, 5(3), Fall 1986, pp 27-52.

[29] R. Bajcsy and F. Solina,
"Three-Dimensional Object Representation Revisited",
IEEE Int. Conf. Computer Vision, 1987, pp 231-240.

[30] B.K.P. Horn,
"Extended Gaussian Images",
Proc. IEEE, 72(12), Dec. 1984, pp 1671-1686.

[31] C.H. Chien, Y.B. Sim and J.K. Aggarwal,
"Generation of Volume/Surface Octree from Range Data",
Proc. IEEE Conf. on Computer Vision and Pattern Recognition,
1988, pp 254-260.

[32] R.C. Bolles and P. Horaud,
"3-DPO : A Three Dimensional Part Orientation System",
Int. Journal of Robotics Research, 5(3), Fall 1986, pp 3-26.

[33] W.E.L. Grimson and T. Lozano-Perez,
"Localizing Overlapping Parts by Searching the Interpretation Tree",
IEEE Trans. Pattern Analysis and Machine Intelligence, 9(4).
July 1987, pp 469-482.

[34] G. Stockman,
"Object Recognition and Localization via Pose Clustering",
Computer Vision, Graphics and Image Processing, 40, 1987, pp 361-387.

[35] C. Goad,
"Special Purpose Automatic Programming for 3D Model-based Vision",
Proc. DARPA I. U. Workshop, 1983.

[36] K. Ikeuchi,
"Generating an Interpretation Tree from a CAD Model for 3-D Object
Recognition in Bin-Picking Tasks",
Int. Jour. Computer Vision, 1987, pp 145-165.

[37] C.V. Ramamoorthy and P.C. Sheu,
"Object-Oriented Systems",
IEEE Expert, Fall 1988, pp 9-15.

[38] C.T. Kitzmiller and J.S. Kowalik,
"Symbolic and Numerical Computing in Knowledge-Based Systems",
CSNCES I, 1986, pp 3-15.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C^3I systems. The areas of technical competence include communications, command and control, battle management information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic reliability/maintainability and compatibility.